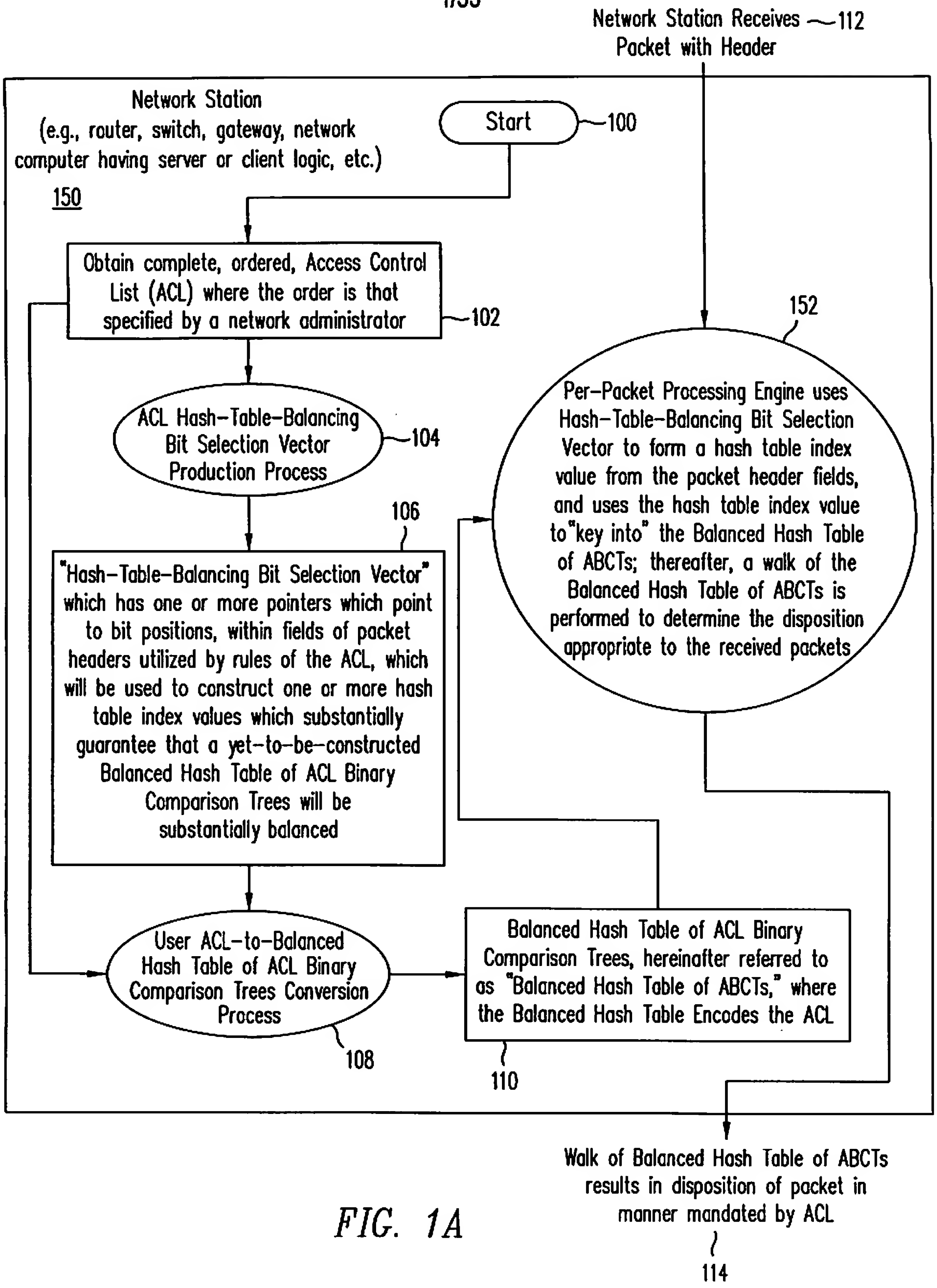

Network Station Receives Packet with Header
112
Network Station
(e.g., router, switch, gateway, network computer having server or client logic, etc.)
150
Start
100
Obtain complete, ordered, Access Control List (ACL) where the order is that specified by a network administrator
102
ACL Hash-Table-Balancing Bit Selection Vector Production Process
104
106
"Hash-Table-Balancing Bit Selection Vector" which has one or more pointers which point to bit positions, within fields of packet headers utilized by rules of the ACL, which will be used to construct one or more hash table index values which substantially guarantee that a yet-to-be-constructed Balanced Hash Table of ACL Binary Comparison Trees will be substantially balanced
152
Per-Packet Processing Engine uses Hash-Table-Balancing Bit Selection Vector to form a hash table index value from the packet header fields, and uses the hash table index value to "key into" the Balanced Hash Table of ABCTs; thereafter, a walk of the Balanced Hash Table of ABCTs is performed to determine the disposition appropriate to the received packets
User ACL-to-Balanced Hash Table of ACL Binary Comparison Trees Conversion Process
108
Balanced Hash Table of ACL Binary Comparison Trees, hereinafter referred to as "Balanced Hash Table of ABCTs," where the Balanced Hash Table Encodes the ACL
110
Walk of Balanced Hash Table of ABCTs results in disposition of packet in manner mandated by ACL
114

FIG. 1A

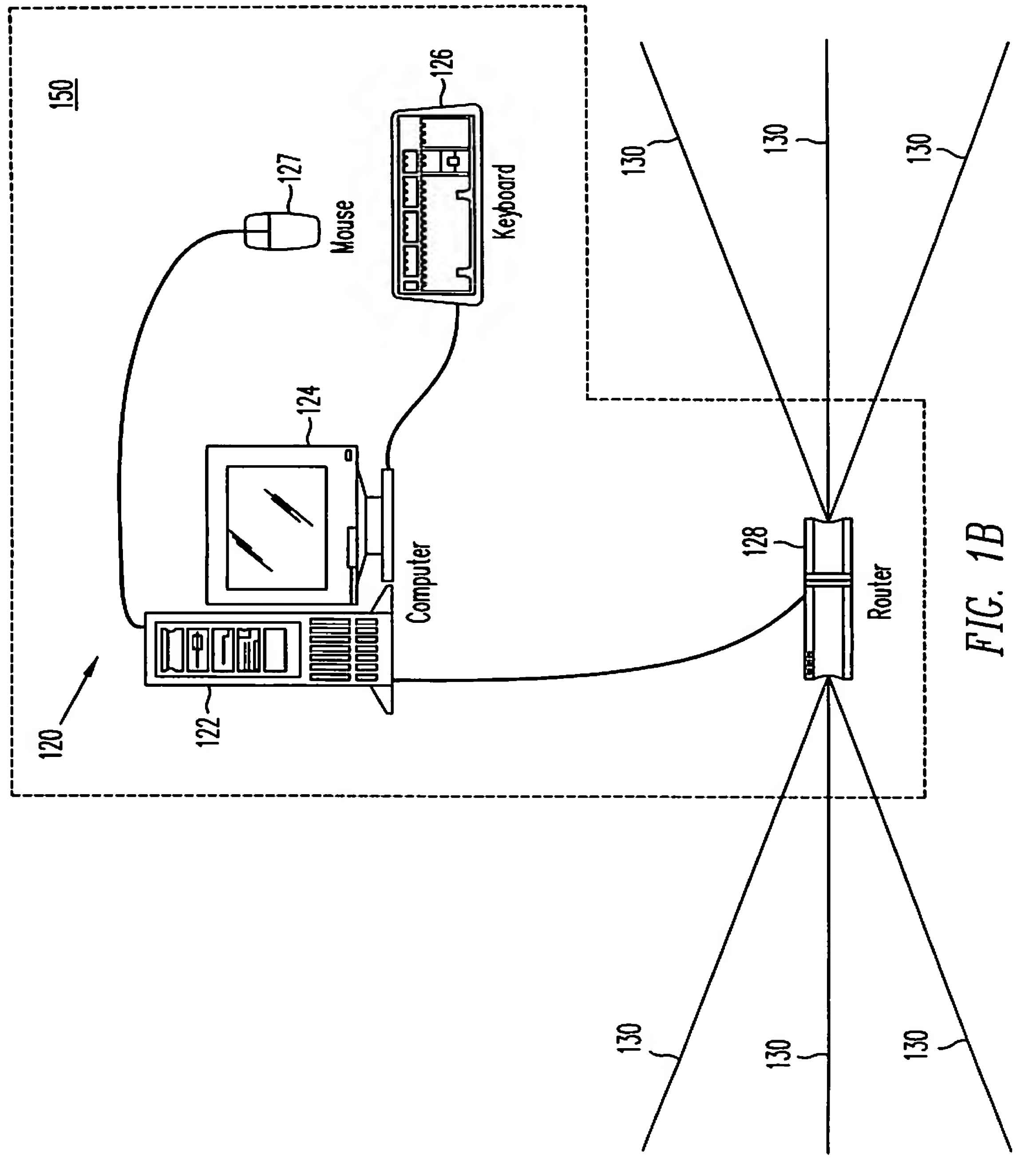

150
127
Mouse
126
Keyboard
124
Computer
120
122
128
Router
130
130
130
130
130
130

FIG. 1B

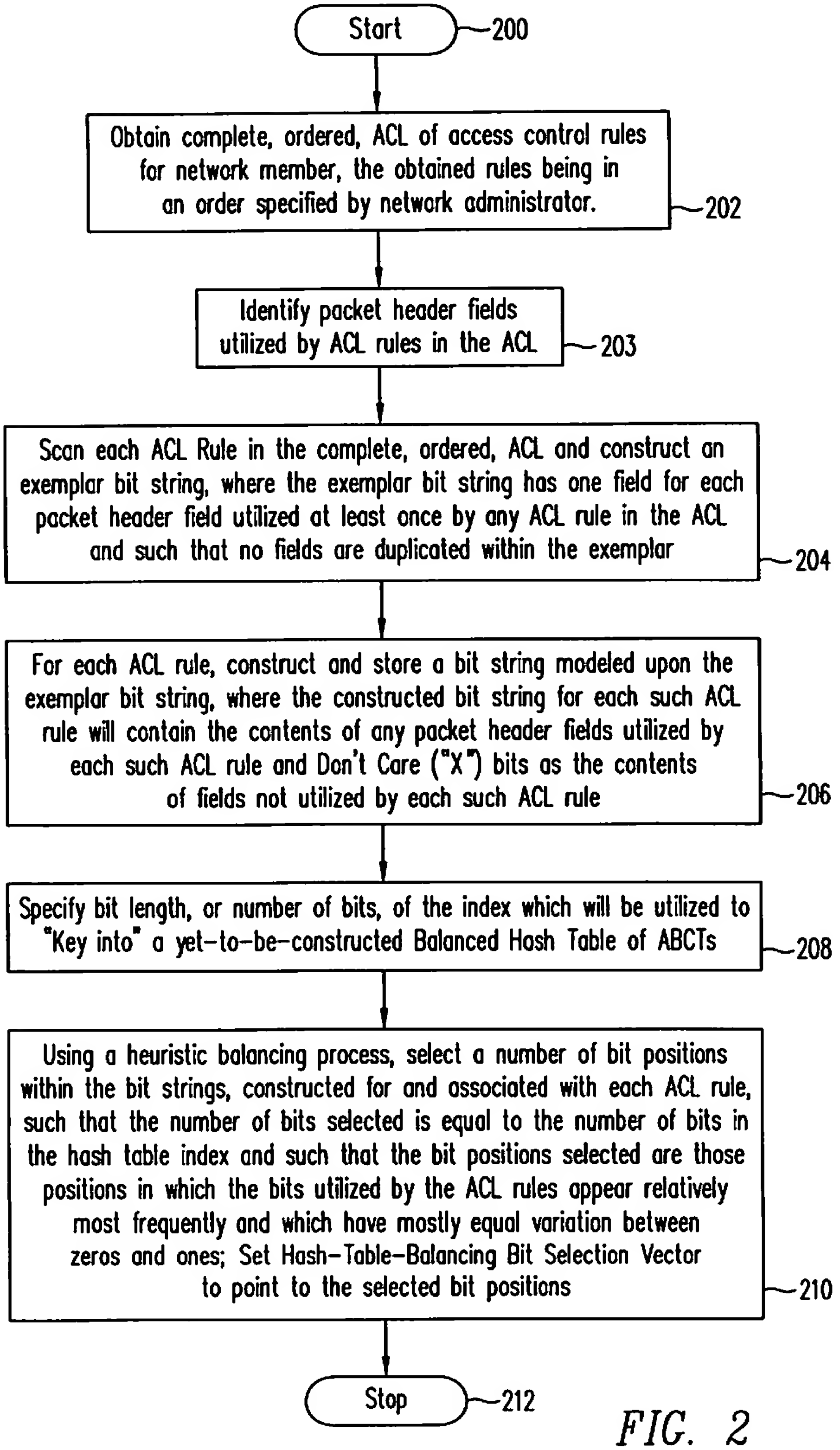
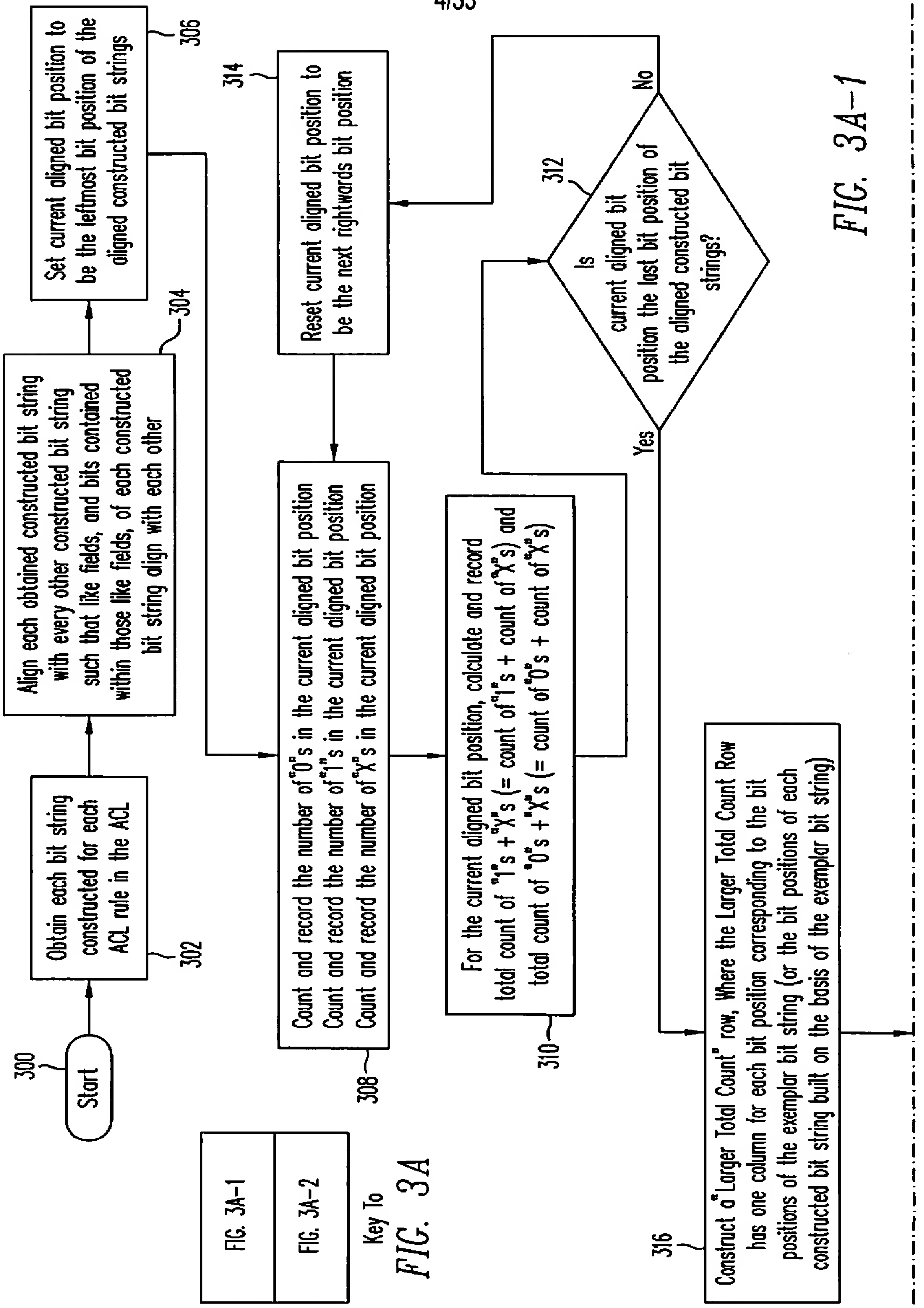

Start
200
Obtain complete, ordered, ACL of access control rules for network member, the obtained rules being in an order specified by network administrator.
202
Identify packet header fields utilized by ACL rules in the ACL
203
Scan each ACL Rule in the complete, ordered, ACL and construct an exemplar bit string, where the exemplar bit string has one field for each packet header field utilized at least once by any ACL rule in the ACL and such that no fields are duplicated within the exemplar
204
For each ACL rule, construct and store a bit string modeled upon the exemplar bit string, where the constructed bit string for each such ACL rule will contain the contents of any packet header fields utilized by each such ACL rule and Don't Care ("X") bits as the contents of fields not utilized by each such ACL rule
206
Specify bit length, or number of bits, of the index which will be utilized to "Key into" a yet-to-be-constructed Balanced Hash Table of ABCTs
208
Using a heuristic balancing process, select a number of bit positions within the bit strings, constructed for and associated with each ACL rule, such that the number of bits selected is equal to the number of bits in the hash table index and such that the bit positions selected are those positions in which the bits utilized by the ACL rules appear relatively most frequently and which have mostly equal variation between zeros and ones; Set Hash-Table-Balancing Bit Selection Vector to point to the selected bit positions
210
Stop
212

FIG. 2


300
Start
302
Obtain each bit string constructed for each ACL rule in the ACL
304
Align each obtained constructed bit string with every other constructed bit string such that like fields, and bits contained within those like fields, of each constructed bit string align with each other
306
Set current aligned bit position to be the leftmost bit position of the aligned constructed bit strings
308
Count and record the number of "0"s in the current aligned bit position
Count and record the number of "1"s in the current aligned bit position
Count and record the number of "X"s in the current aligned bit position
310
For the current aligned bit position, calculate and record total count of "1"s + "X"s (= count of "1"s + count of "X"s) and total count of "0"s + "X"s (= count of "0"s + count of "X"s)
312
Is current aligned bit position the last bit position of the aligned constructed bit strings?
Yes
No
314
Reset current aligned bit position to be the next rightwards bit position
316
Construct a "Larger Total Count" row, Where the Larger Total Count Row has one column for each bit position corresponding to the bit positions of the exemplar bit string (or the bit positions of each constructed bit string built on the basis of the exemplar bit string)
FIG. 3A-1
FIG. 3A-2

Key To
FIG. 3A

FIG. 3A-1

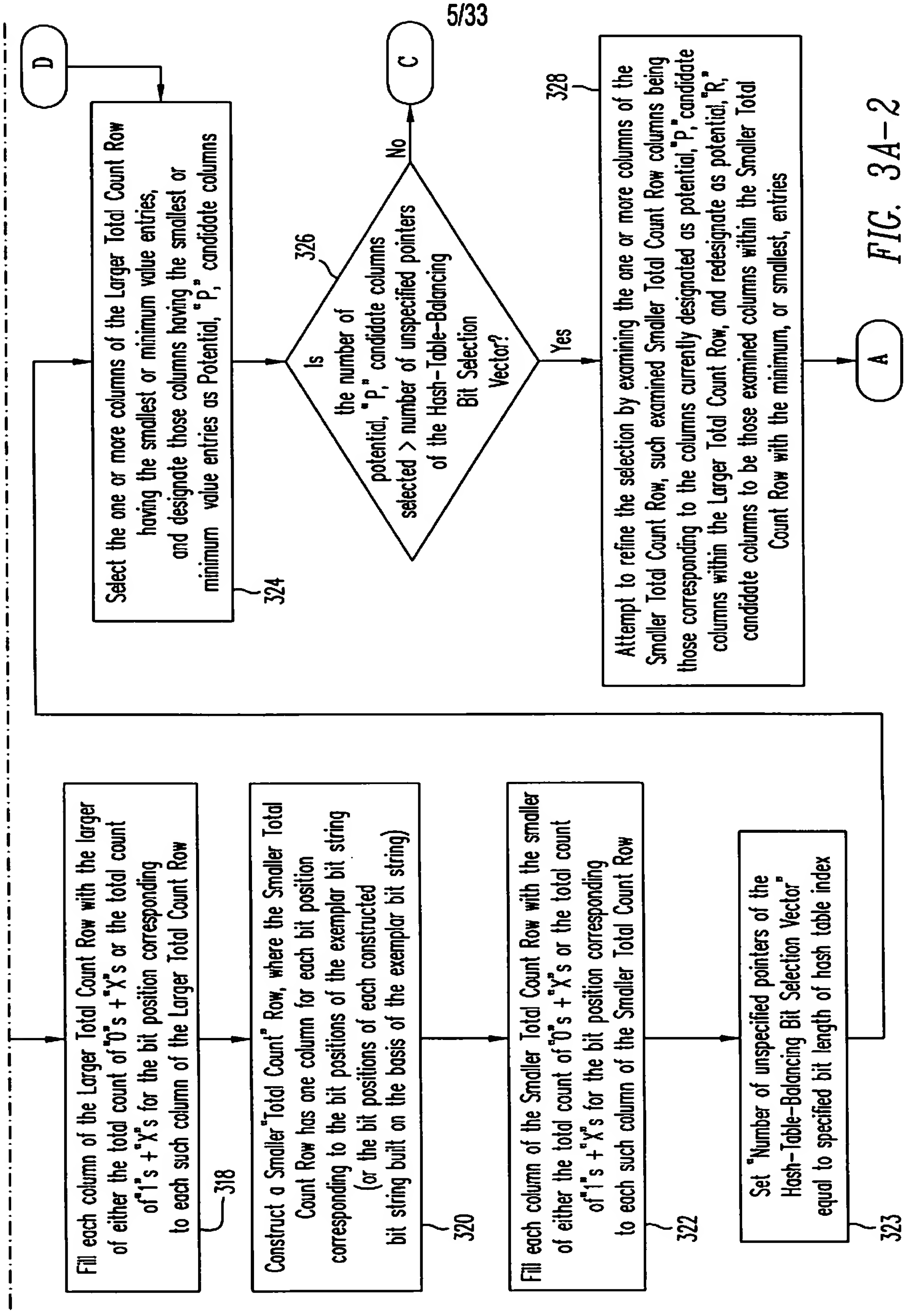

Fill each column of the Larger Total Count Row with the larger of either the total count of "0"s + "X"s or the total count of "1"s + "X"s for the bit position corresponding to each such column of the Larger Total Count Row
318
Construct a Smaller "Total Count" Row, where the Smaller Total Count Row has one column for each bit position corresponding to the bit positions of the exemplar bit string (or the bit positions of each constructed bit string built on the basis of the exemplar bit string)
320
Fill each column of the Smaller Total Count Row with the smaller of either the total count of "0"s + "X"s or the total count of "1"s + "X"s for the bit position corresponding to each such column of the Smaller Total Count Row
322
Set "Number of unspecified pointers of the Hash-Table-Balancing Bit Selection Vector" equal to specified bit length of hash table index
323
D
Select the one or more columns of the Larger Total Count Row having the smallest or minimum value entries, and designate those columns having the smallest or minimum value entries as Potential, "P," candidate columns
324
Is the number of potential, "P," candidate columns selected > number of unspecified pointers of the Hash-Table-Balancing Bit Selection Vector?
326
No
C
Yes
Attempt to refine the selection by examining the one or more columns of the Smaller Total Count Row, such examined Smaller Total Count Row columns being those corresponding to the columns currently designated as potential, "P," candidate columns within the Larger Total Count Row, and redesignate as potential, "R," candidate columns to be those examined columns within the Smaller Total Count Row with the minimum, or smallest, entries
328
A

FIG. 3A-2

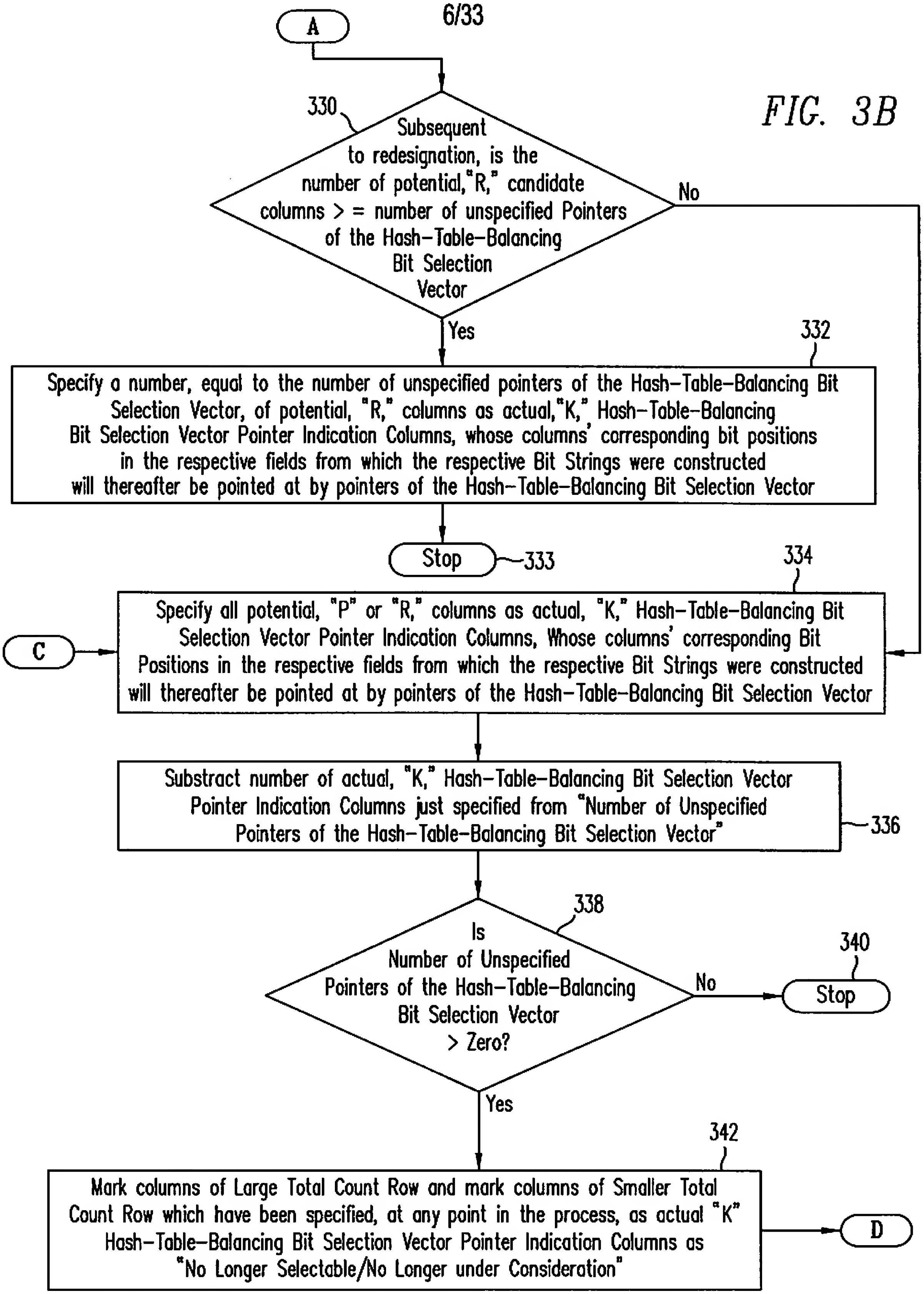

A
330
Subsequent to redesignation, is the number of potential,"R," candidate columns > = number of unspecified Pointers of the Hash-Table-Balancing Bit Selection Vector
No
Yes
332
Specify a number, equal to the number of unspecified pointers of the Hash-Table-Balancing Bit Selection Vector, of potential, "R," columns as actual,"K," Hash-Table-Balancing Bit Selection Vector Pointer Indication Columns, whose columns' corresponding bit positions in the respective fields from which the respective Bit Strings were constructed will thereafter be pointed at by pointers of the Hash-Table-Balancing Bit Selection Vector
Stop
333
334
C
Specify all potential, "P" or "R," columns as actual, "K," Hash-Table-Balancing Bit Selection Vector Pointer Indication Columns, Whose columns' corresponding Bit Positions in the respective fields from which the respective Bit Strings were constructed will thereafter be pointed at by pointers of the Hash-Table-Balancing Bit Selection Vector
Substract number of actual, "K," Hash-Table-Balancing Bit Selection Vector Pointer Indication Columns just specified from "Number of Unspecified Pointers of the Hash-Table-Balancing Bit Selection Vector"
336
338
Is Number of Unspecified Pointers of the Hash-Table-Balancing Bit Selection Vector > Zero?
No
340
Stop
Yes
342
Mark columns of Large Total Count Row and mark columns of Smaller Total Count Row which have been specified, at any point in the process, as actual "K" Hash-Table-Balancing Bit Selection Vector Pointer Indication Columns as "No Longer Selectable/No Longer under Consideration"
D

FIG. 3B

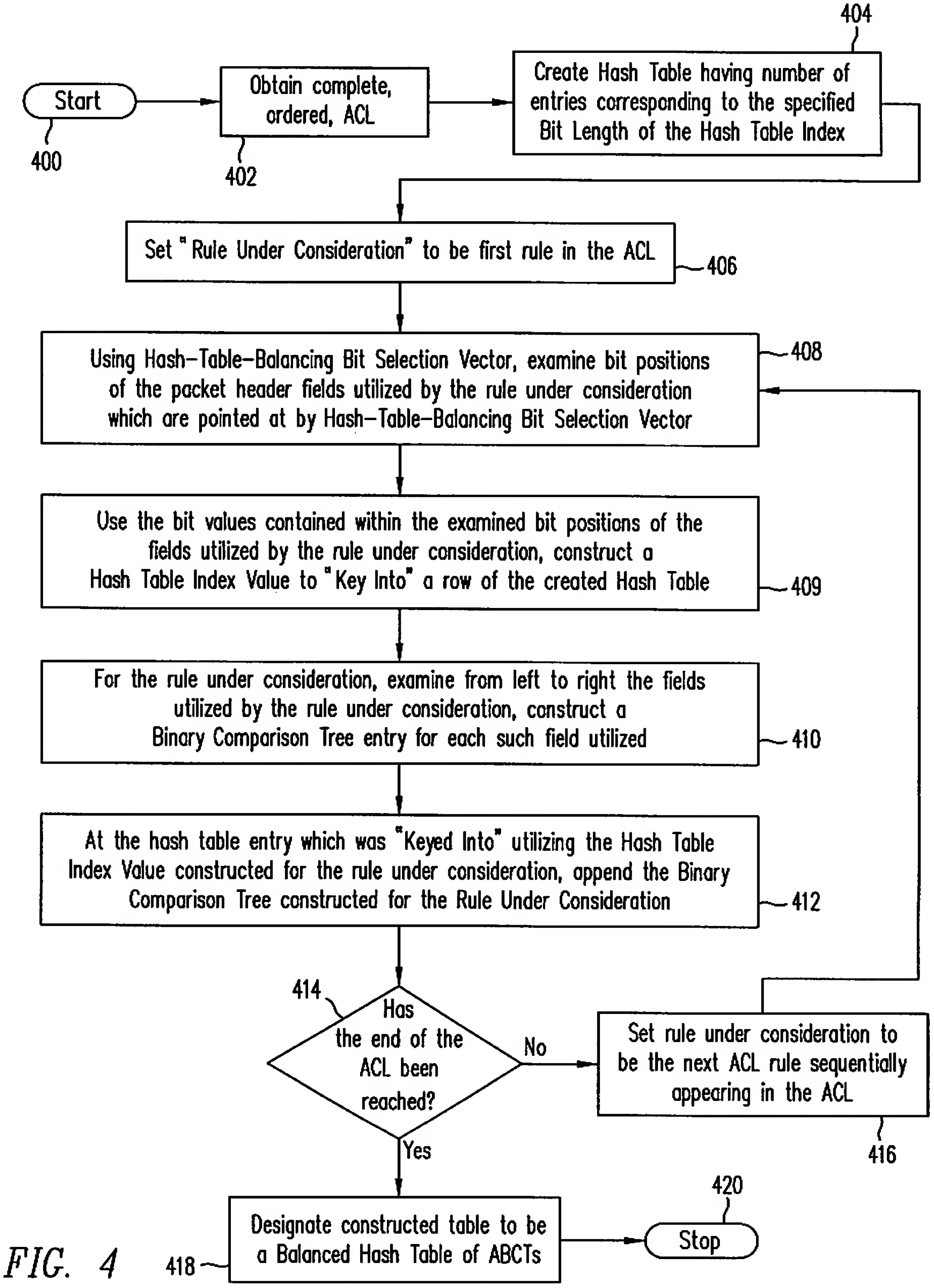

Start
400
Obtain complete, ordered, ACL
402
Create Hash Table having number of entries corresponding to the specified Bit Length of the Hash Table Index
404
Set "Rule Under Consideration" to be first rule in the ACL
406
Using Hash-Table-Balancing Bit Selection Vector, examine bit positions of the packet header fields utilized by the rule under consideration which are pointed at by Hash-Table-Balancing Bit Selection Vector
408
Use the bit values contained within the examined bit positions of the fields utilized by the rule under consideration, construct a Hash Table Index Value to "Key Into" a row of the created Hash Table
409
For the rule under consideration, examine from left to right the fields utilized by the rule under consideration, construct a Binary Comparison Tree entry for each such field utilized
410
At the hash table entry which was "Keyed Into" utilizing the Hash Table Index Value constructed for the rule under consideration, append the Binary Comparison Tree constructed for the Rule Under Consideration
412
414
Has the end of the ACL been reached?
No
Set rule under consideration to be the next ACL rule sequentially appearing in the ACL
416
Yes
Designate constructed table to be a Balanced Hash Table of ABCTs
418
Stop
420

FIG. 4

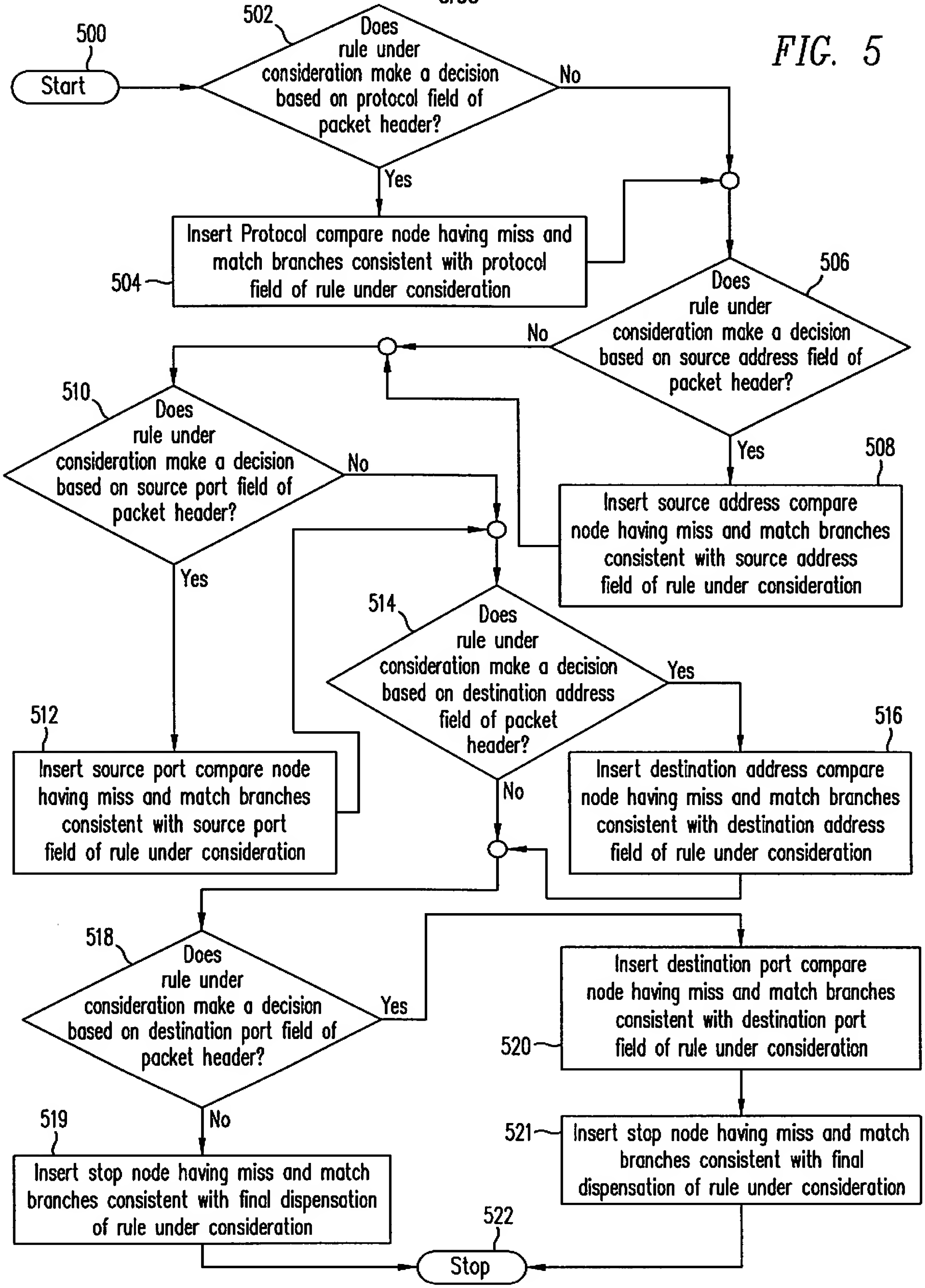

500
Start
502
Does rule under consideration make a decision based on protocol field of packet header?
No
Yes
504
Insert Protocol compare node having miss and match branches consistent with protocol field of rule under consideration
506
Does rule under consideration make a decision based on source address field of packet header?
No
Yes
508
Insert source address compare node having miss and match branches consistent with source address field of rule under consideration
510
Does rule under consideration make a decision based on source port field of packet header?
No
Yes
512
Insert source port compare node having miss and match branches consistent with source port field of rule under consideration
514
Does rule under consideration make a decision based on destination address field of packet header?
Yes
No
516
Insert destination address compare node having miss and match branches consistent with destination address field of rule under consideration
518
Does rule under consideration make a decision based on destination port field of packet header?
Yes
No
520
Insert destination port compare node having miss and match branches consistent with destination port field of rule under consideration
519
Insert stop node having miss and match branches consistent with final dispensation of rule under consideration
521
Insert stop node having miss and match branches consistent with final dispensation of rule under consideration
522
Stop

FIG. 5

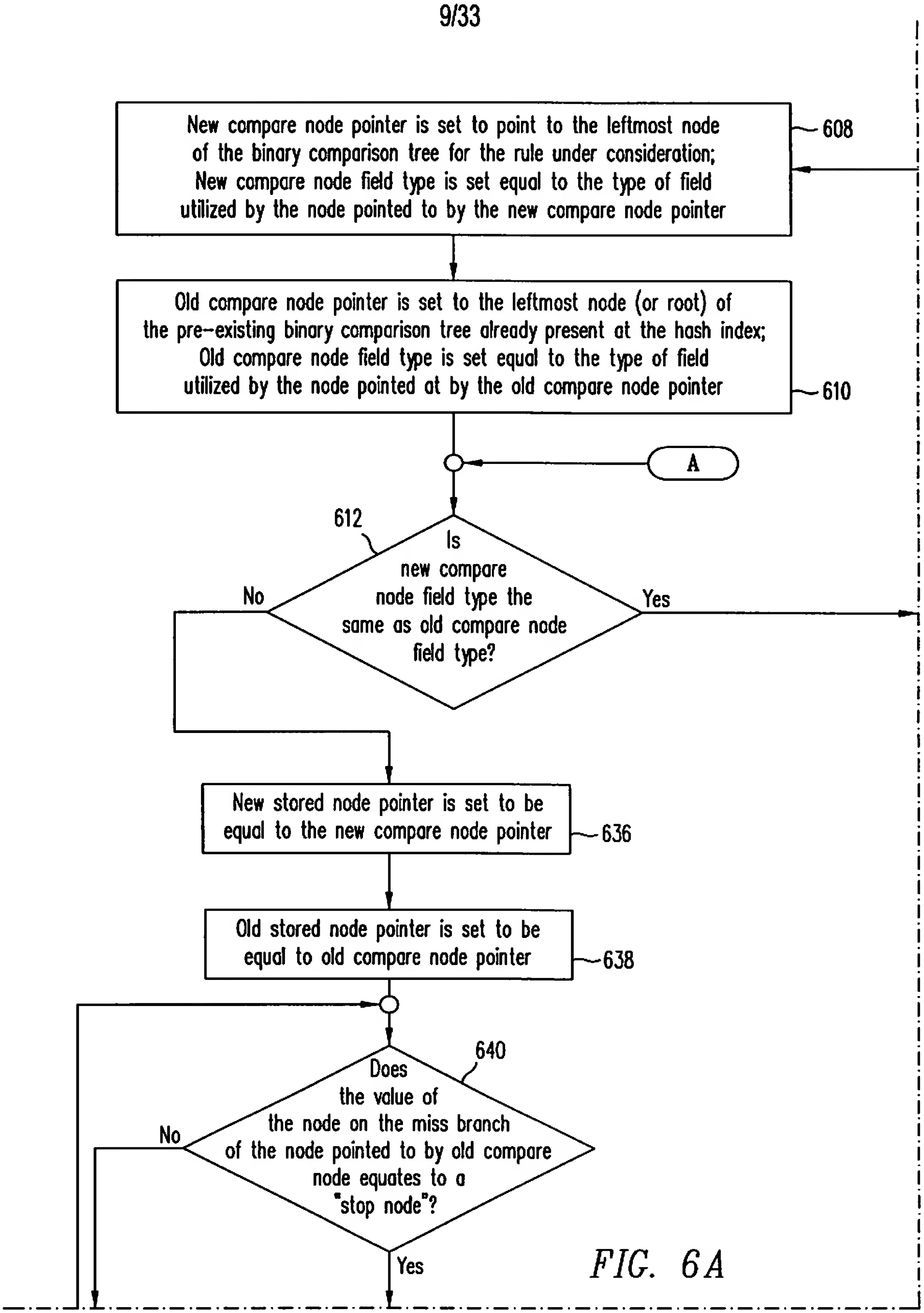

New compare node pointer is set to point to the leftmost node of the binary comparison tree for the rule under consideration; New compare node field type is set equal to the type of field utilized by the node pointed to by the new compare node pointer
608
Old compare node pointer is set to the leftmost node (or root) of the pre-existing binary comparison tree already present at the hash index; Old compare node field type is set equal to the type of field utilized by the node pointed at by the old compare node pointer
610
A
612
Is new compare node field type the same as old compare node field type?
No
Yes
New stored node pointer is set to be equal to the new compare node pointer
636
Old stored node pointer is set to be equal to old compare node pointer
638
640
Does the value of the node on the miss branch of the node pointed to by old compare node equates to a "stop node"?
No
Yes

FIG. 6A

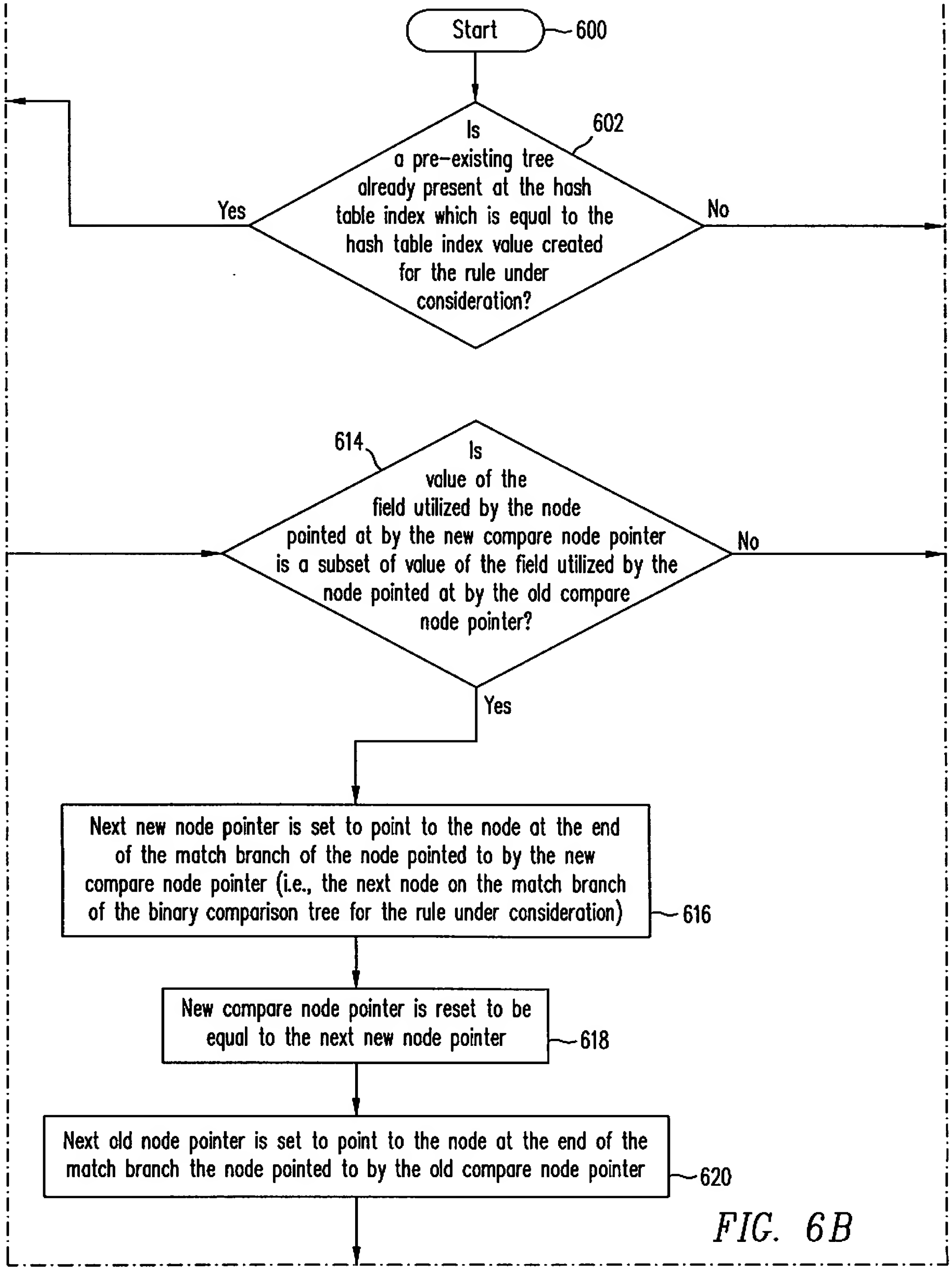

Start
600
Is a pre-existing tree already present at the hash table index which is equal to the hash table index value created for the rule under consideration?
602
Yes
No
614
Is value of the field utilized by the node pointed at by the new compare node pointer is a subset of value of the field utilized by the node pointed at by the old compare node pointer?
No
Yes
Next new node pointer is set to point to the node at the end of the match branch of the node pointed to by the new compare node pointer (i.e., the next node on the match branch of the binary comparison tree for the rule under consideration)
616
New compare node pointer is reset to be equal to the next new node pointer
618
Next old node pointer is set to point to the node at the end of the match branch the node pointed to by the old compare node pointer
620

FIG. 6B

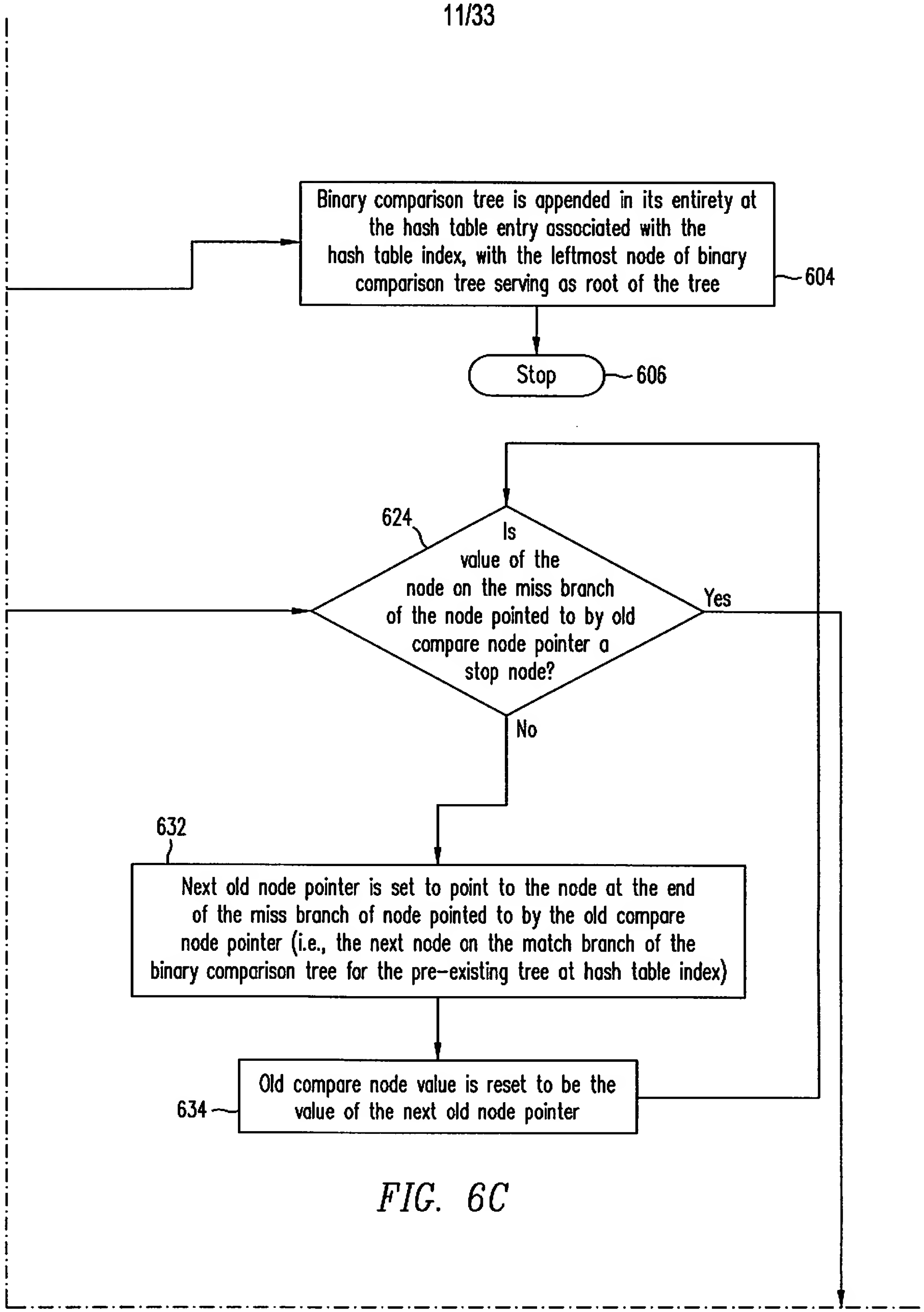

Binary comparison tree is appended in its entirety at the hash table entry associated with the hash table index, with the leftmost node of binary comparison tree serving as root of the tree
604
Stop
606
624
Is value of the node on the miss branch of the node pointed to by old compare node pointer a stop node?
Yes
No
632
Next old node pointer is set to point to the node at the end of the miss branch of node pointed to by the old compare node pointer (i.e., the next node on the match branch of the binary comparison tree for the pre-existing tree at hash table index)
634
Old compare node value is reset to be the value of the next old node pointer

FIG. 6C

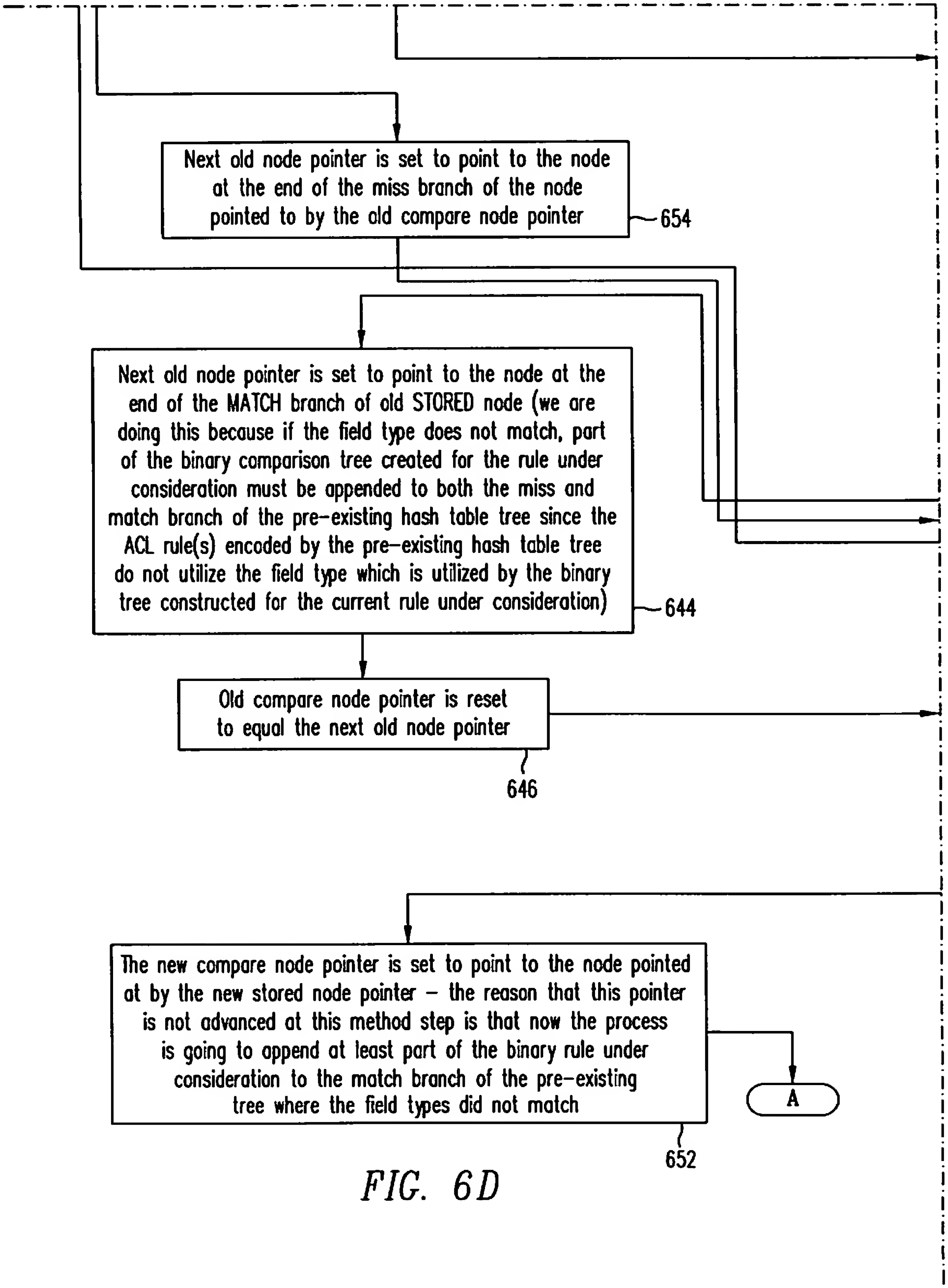

Next old node pointer is set to point to the node at the end of the miss branch of the node pointed to by the old compare node pointer
654
Next old node pointer is set to point to the node at the end of the MATCH branch of old STORED node (we are doing this because if the field type does not match, part of the binary comparison tree created for the rule under consideration must be appended to both the miss and match branch of the pre-existing hash table tree since the ACL rule(s) encoded by the pre-existing hash table tree do not utilize the field type which is utilized by the binary tree constructed for the current rule under consideration)
644
Old compare node pointer is reset to equal the next old node pointer
646
The new compare node pointer is set to point to the node pointed at by the new stored node pointer – the reason that this pointer is not advanced at this method step is that now the process is going to append at least part of the binary rule under consideration to the match branch of the pre-existing tree where the field types did not match
A
652

FIG. 6D

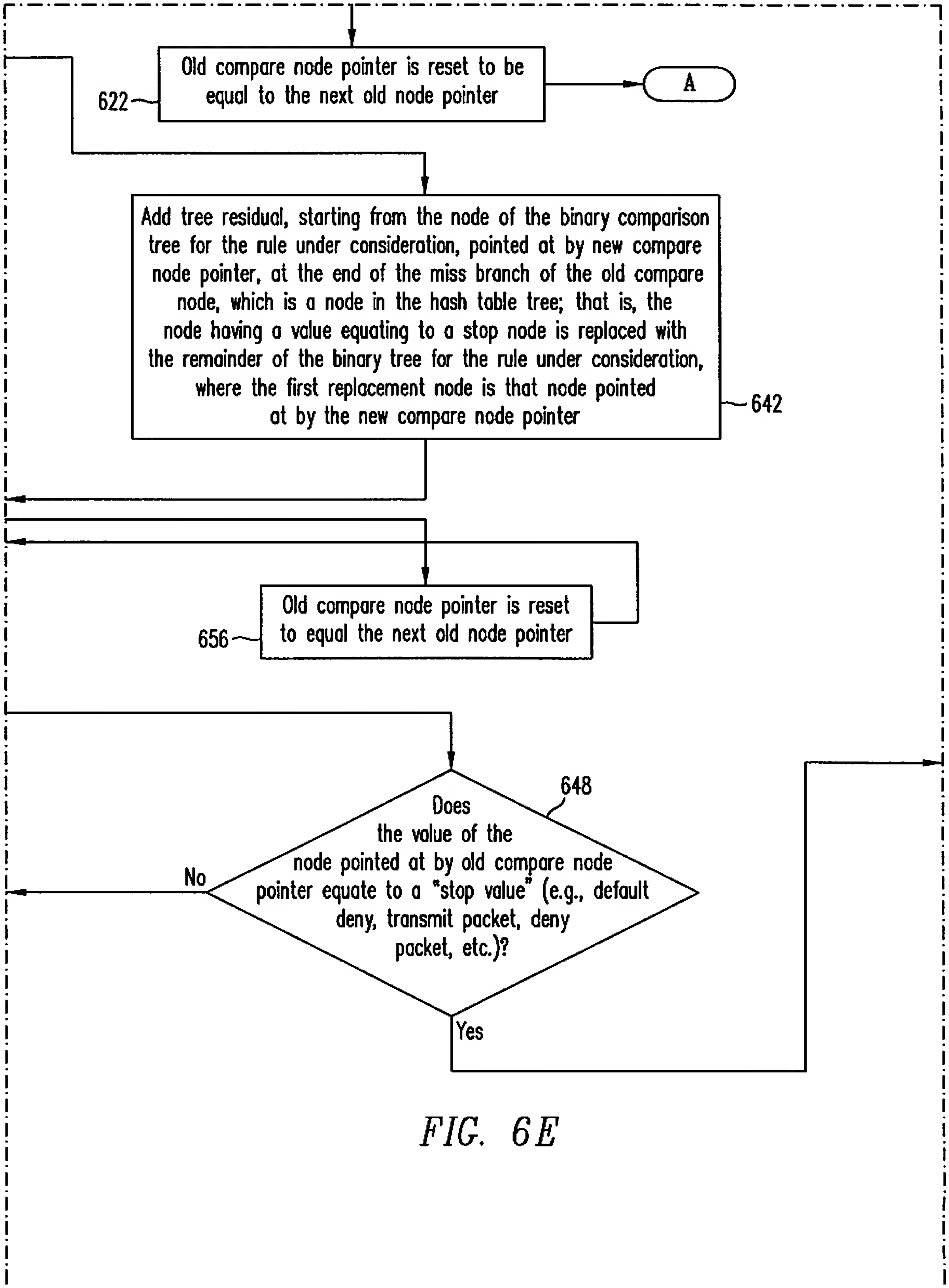

Old compare node pointer is reset to be equal to the next old node pointer
622
A
Add tree residual, starting from the node of the binary comparison tree for the rule under consideration, pointed at by new compare node pointer, at the end of the miss branch of the old compare node, which is a node in the hash table tree; that is, the node having a value equating to a stop node is replaced with the remainder of the binary tree for the rule under consideration, where the first replacement node is that node pointed at by the new compare node pointer
642
Old compare node pointer is reset to equal the next old node pointer
656
Does the value of the node pointed at by old compare node pointer equate to a "stop value" (e.g., default deny, transmit packet, deny packet, etc.)?
648
No
Yes

FIG. 6E

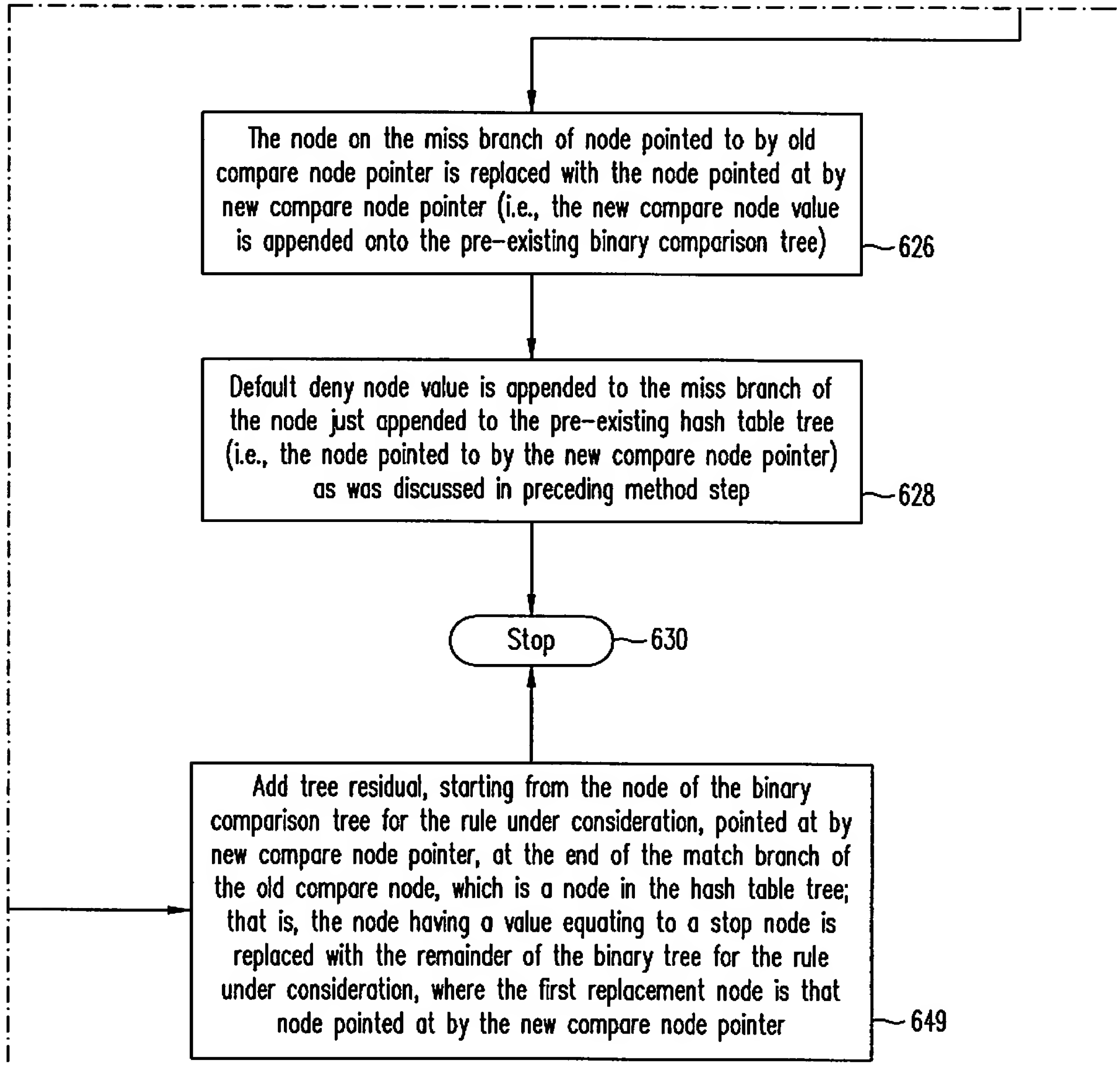
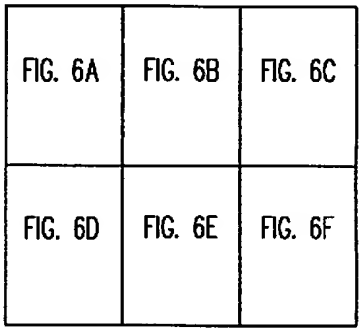

The node on the miss branch of node pointed to by old compare node pointer is replaced with the node pointed at by new compare node pointer (i.e., the new compare node value is appended onto the pre-existing binary comparison tree)
626
Default deny node value is appended to the miss branch of the node just appended to the pre-existing hash table tree (i.e., the node pointed to by the new compare node pointer) as was discussed in preceding method step
628
Stop
630
Add tree residual, starting from the node of the binary comparison tree for the rule under consideration, pointed at by new compare node pointer, at the end of the match branch of the old compare node, which is a node in the hash table tree; that is, the node having a value equating to a stop node is replaced with the remainder of the binary tree for the rule under consideration, where the first replacement node is that node pointed at by the new compare node pointer
649

FIG. 6F



Key To
FIG. 6

Simplified Example of Ordered ACL Rule Set Typically Entered by a Network Administrator

ACL Rules in an Ordered ACL Rule Set expressed as plain english statements	Examples of Coded Versions of ACL Rules Which Are Typically Utilized Within an ACL Rule Set
Permit TCP protocol packets from any source IP address going to host having an IP address of 28.16.31.10 and a port identifier equal to 28.	PERMIT TCP ANY HOST 28.16.31.10 EQ 28
Deny TCP protocol packets from any source IP address going to host having an IP address of 28.16.31.10 and a port identifier greater than 23.	DENY TCP ANY HOST 28.16.31.10 GT 23
Deny UDP protocol packets from any source IP address going to host having an IP address of 30.22.12.5 and a port identifier equal to 11.	DENY UDP ANY HOST 30.22.21.5 EQ 11
Permit UDP protocol packets from any source IP address going to host having an IP address of 30.22.12.X, where X indicates any number, or "don't care".	PERMIT UDP ANY HOST 30.22.21.X
Deny all packets having source IP address of 23.20.7.0 and any destination address (indicated by address X.X.X.X, where X indicates any number, or "don't care").	DENY TCP 23.20.7.0 X.X.X.X.
Permit TCP protocol packets from any source IP address going to host having an IP address of 28.16.32.10.	PERMIT TCP ANY HOST 28.16.31.10

FIG. 7A

Example of the Creation of an Exemplar Bit String Having One Field for Each Packet Header Field Utilized by at Least One ACL Rule in the ACL Rule Set, and the Subsequent Creation of Bit Strings for each ACL Rule in the ACL Rule Set Based on the Created Exemplar

Construct Exemplar Bit String Based On Packet Header Fields Utilized by ACL Rule Set Rules	Protocol ID Source Address Destination Address Destination Port
Bit String, based on exemplar, for ACL Rule 1 with string "01001" associated with TCP protocol for sake of example.	01001.XXXXX.XXXXX.XXXXX.XXXXX.11100.10000.11111.01010.10111
Bit String, based on exemplar, for ACL Rule 2 with string "01001" associated with TCP protocol for sake of example.	01001.XXXXX.XXXXX.XXXXX.XXXXX.11100.10000.11111.01010.11100
Bit String, based on exemplar, for ACL Rule 3 with string "11111" associated with UDP protocol for sake of example.	11111.XXXXX.XXXXX.XXXXX.XXXXX.11110.10110.10101.00101.01011
Bit String, based on exemplar, for ACL Rule 4 with string "11111" associated with UDP protocol for sake of example.	11111.XXXXX.XXXXX.XXXXX.XXXXX.11110.10110.10101.XXXXX.XXXXX
Bit String, based on exemplar, for ACL Rule 5 with string "01001" associated with TCP protocol for sake of example.	01001.10111.10101.00111.00000.XXXXX.XXXXX.XXXXX.XXXXX.XXXXX
Bit String, based on exemplar, for ACL Rule 6 with string "01001" associated with TCP protocol for sake of example.	01001.XXXXX.XXXXX.XXXXX.XXXXX.11100.10000.11111.01010.XXXXX
Note: "Bit Position" is illustrated for sake of clarity and ease of counting herein as taking account of the periods between 5 bit fields; however, those skilled in the art will recognize that ordinarily such periods are not counted as bit positions.	00000000011111111112222222222333333333344444444445555555555 ←— Read Bit Position 12345678901234567890123456789012345678901234567890123456789 ←— Slot Numbers Vertically. For example, the first bit position is denoted 0 0 0 0 1, the second, 2, the third, 3, the fourth, 4, 1 5 the eleventh, 1, and the fifty-ninth, 9.

FIG. 7B

Example of the Creation of a Bit Selection Vector

"0" Count in Each Bit Postion: "X" Count in Each Bit Postion:	40440.01000.01010.11000.11111.00035.05335.02020.41313.11111 00000.55555.55555.55555.55555.11111.11111.11111.22222.33333
Total of "0" + "X" Counts:	40440.56555.56565.66555.66666.11146.16446.13131.63535.44444
"1" Count in Each Bit Postion: "X" Count in Each Bit Postion:	26226.10111.10101.00111.00000.55520.50220.53535.03131.22222 00000.55555.55555.55555.55555.11111.11111.11111.22222.33333
Total of "1" + "X" Counts:	26226.65666.65656.55666.55555.66631.61331.64645.25353.55555
Construct a "Larger Total Count" row having one row entry corresponding to each bit position in the strings which were constructed from the ACL rules; fill each row entry with the larger of either the "Total of '0' + 'X' Counts" or "Total of '1' + 'X' Counts" for the bit position corresponding to that row entry.	46446.66666.66666.66666.66666.66646.66446.64645.65555.55555
Construct a "Smaller Total Count" row having one row entry corresponding to each bit position in the strings which were constructed from the ACL rules; fill each row entry with the smaller of either the "Total of '0' + 'X' Counts" or "Total of '1' + 'X' Counts" for the bit position corresponding to that row entry.	20226.55555.55555.55555.55555.11131.11331.13131.23333.44444
Set number of unspecified pointers of bit selection vector = specified bit length of hash table index	Number of Unspecified Pointers of Bit Selection Vector = 4 For sake of example, assume hash table index having a bit length of 4 is specified.
Select the row entries in the "Larger Total Count" Row columns having the smallest number entries; designate the bit positions corresponding to the selected row columns as potential, "P," candidate columns which might be utilized as the pointers of the bit selection vector	P PP P PP P P Note: The row columns 1, 3, 34, 39, 41, and 46 of the "Larger Total Count" row had the smallest entries (i.e., the base 10 number "4"), and thus the bit positions associated with row columns 1, 3, 34, 39, 41, and 46 of the "Larger Total Count" row are designated as potential candidate bits "P."
Since there are more potential, "P," candidate columns than number of unspecified pointers of bit selection vector, refine the selection by examining the columns of the Smaller Total	R RR Note: The row columns 1, 3, and 4 of the "Smaller Total Count" row, corresponding with the selected row columns of the "Larger Total Count" row, had the smallest entries (i.e., the base 10 number "2"), and thus the bit positions associated with row columns 1, 3, and 4 of the "Smaller Total Count" row are redesignated as

FIG. 7C1

Count Row, with such examined Smaller Total Count Row columns being those corresponding to the Larger Total Count Row columns designated as potential,"P," candidate columns; redesignate as potential,"R," candidate columns which might be utilized as the pointers of the bit selection vector, those examined Smaller Total Count Row columns with the smallest number entries	candidate bits "R".
Since the number or redesignated potential candidates, "R," is less than the number of unspecified pointers of bit selection vector, designate all redesignated, "R," candidates as actual, "K," bit selection vector Pointer Indication Columns, whose corresponding bit positions in the respective fields from which the respective bit strings were constructed will thereafter be pointed at by pointers of the bit selection vector	K KK Note: The number of unspecified pointers of bit selection vector is currently equal to 4, and the number of redesignated potential candidates, "R," is 3, which is less than the number of unspecified pointers of bit selection vector; thus, all "R" potential candidates are specified actual, "K," bit selection vector pointer indication columns, whose corresponding bit positions in the respective fields from which the respective bit strings were constructed will thereafter be pointed at by pointers of the bit selection vector.
Substract the number of specified actual, "K," bit selection vector pointer indication columns, whose corresponding bit positions in the respective fields from which the respective bit strings were constructed will thereafter be pointed at by pointers of the bit selection vector, from number of unspecified pointers of bit selection vector	Number of unspecified pointers of bit selection vector = number of unspecified pointers of bit selection vector (i.e., 4) - number of specified actual, "K," bit selection vector pointer indication columns, whose corresponding bit positions in the respective fields from which the repective bit strings were constructed will thereafter be pointed at by pointers of the bit selection vector specified in preceding step (i.e., 3) = 1 pointer left unspecified
Since the number of unspecified pointers of bit selection vector is still non-zero, mark specified "K," bit selection vector pointer indication columns, whose corresponding bit positions in the respective fields from which the respective bit strings were constructed will thereafter be pointed at by	* ** Note: Row columns 1, 3, and 4 are marked with asterisks to indicate that since these row columns have already been designated as candidates "K," bit selection vector pointer indication columns, whose corresponding bit positions in the respective fields from which the respective bit strings were constructed will thereafter be pointed at by pointers of the bit selection vector.

FIG. 7C2

pointers of the bit selection vector with asterisks indicating that such columns are no longer selectable or under consideration, since the bit positions associated with the "K," bit selection vector pointer indication columns, whose corresponding bit positions in the respective fields from which the respective bit strings were constructed will thereafter be pointed at by pointers of the bit selection vector have already been specified	
Thereafter, repeat the "select the row entries in the Larger Total Count" Row having smallest number entries . . . " operation above upon the row columns which have not yet been designated as candidate "K," bit selection vector pointer indication columns, whose corresponding bit positions in the respective fields from which the respective bit strings were constructed will thereafter be pointed at by pointers of the bit selection vector	* ** P PP P P Note: Row columns 1, 3, and 4 are marked with asterisks to indicate that since the bit positions associated with these row columns have already been designated as candidates.

FIG. 7C3

Since there are more candidates, "P," than number of unspecified pointers of bit selection vector (at this point, 3 pointers have been specified as "K," meaning that one additional pointer is necessary to have the pointers required to completely point out the 4 bit hash table index), repeat the refine the selection operation above	R RR R R Note: Since all entries in the "Smaller Total Count" Row columns, corresponding with the selected row columns of the "Larger Total Count" Row, were the same number (i.e., the base ten number "3"), all P row columns are redesignated as candidates"R".
Since after redesignation there are still more candidates "R" than the number of unspecified pointers of bit selection vector, all "R," candidates are deemed equally good choices; consequentially, the number of actual, "K," bit selection vector pointer indication columns, whose corresponding bit positions in the respective fields from which the respective bit strings were constructed will thereafter be pointed at by pointers of the bit selection vector necessary to completely point out the hash table index value (i.e., in the present example, one more pointer is needed) may be selected at random from the designated "R" row columns.	K Note: Select row column 34 at random.
There are now specified actual, "K," bit selection vector pointer indication columns, whose corresponding bit positions in the respective fields from which the respective bit strings were constructed will thereafter be pointed at by pointers of the bit selection vector equal in number to the bit length of the hash table index; consequently, all pointers of the bit selection vector, which will be utilized to point to bit positions used to form a hash table index value which will be used to "key into"	K KK Note: These actual, "K," bit selection vector pointer indication columns, whose corresponding bit positions in the respective fields from which the respective bit strings were constructed will thereafter be pointed at by pointers of the bit selection vector indicate that the first, third, and fourth leftmost bit positions within the "protocol ID" field, and the fourth leftmost bit positions within the "destination address" field will be utilized as the hash table index bits.

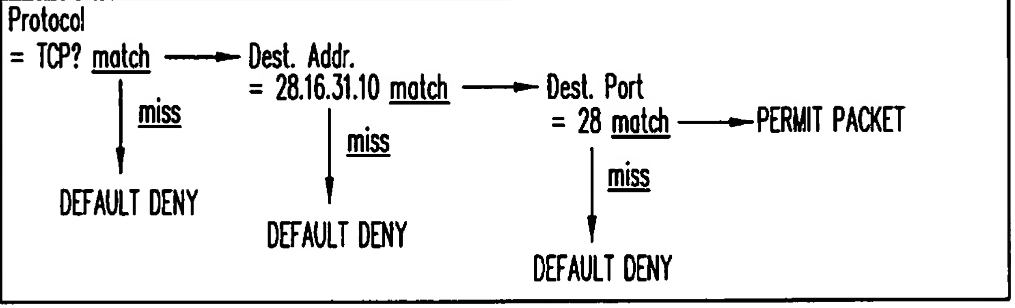
FIG. 7C4

the hash table, have been fully specified.	
Definition of the bit selection vector	Bit Selection Vector = (pointer to first leftmost bit position within the "protocol ID" field; pointer to third leftmost bit position within the "protocol ID" field; pointer to fourth leftmost bit position within the "protocol ID" field; pointer to fourth leftmost bit position within the "destination address" field)

FIG. 7C5

Example showing the construction of Balanced Hash Table of ACL Binary Comparison Trees

Example showing the creation of a Binary Comparison Tree for First In Sequence ACL Rule in Rule Set

PERMIT TCP ANY HOST 28.16.31.10 EQ 28	 Protocol = TCP? match miss DEFAULT DENY Dest. Addr. = 28.16.31.10 match miss DEFAULT DENY Dest. Port = 28 match miss DEFAULT DENY PERMIT PACKET

Example showing the addition of a Binary Comparison Tree constructed for the first in Sequence Rule in ACL Rule Set into the Hash Table

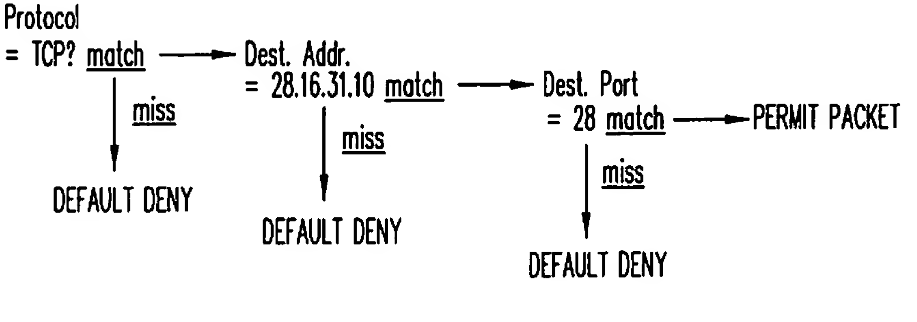
Select bit string constructed from first ACL rule in Rule Set, utilizing the contents of those bit positions (1, 3, 4, and 34) pointed at by the Hash-Table-Balancing Bit Selection Vector, enter hash table at entry corresponding to the bits at bit positions serving as hash key index (e.g., bit position 1 contains "0"; bit position 3 contains"0"; bit position 4 contains"0"; and bit position 34 contains"0") and build binary Comparison Tree indicative of this first selected ACL rule	0000	 Protocol = TCP? match miss DEFAULT DENY Dest. Addr. = 28.16.31.10 match miss DEFAULT DENY Dest. Port = 28 match miss DEFAULT DENY PERMIT PACKET
	0001	
	0010	
	0011	
	0100	
	0101	
	0110	
	0111	
	1000	
	1001	
	1010	

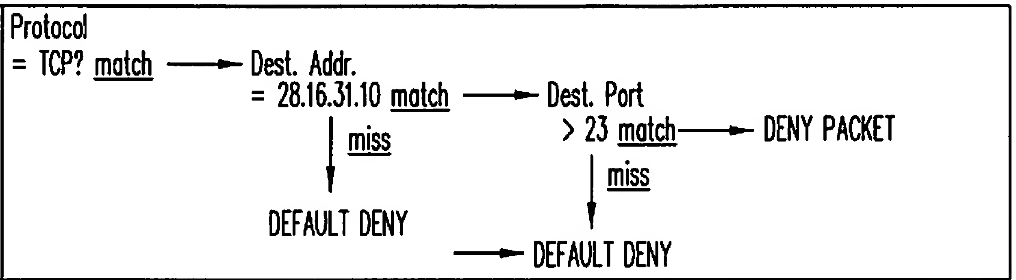
FIG. 7D1

	1011	
	1100	
	1101	
	1110	
	1111	

FIG. 7D2

Example Showing the Construction of Balanced Hash Table of ACL Binary Comparison Trees (cont.)

Example Showing the Creation of a Binary Comparison Tree for Second In Sequence Rule in Rule Set

DENY TCP ANY HOST 28.16.31.10 GT 23	

Example Showing the Addition of a Binary Comparison Tree Constructed for the Second In Sequence Rule in ACL Rule Set into the Hash Table

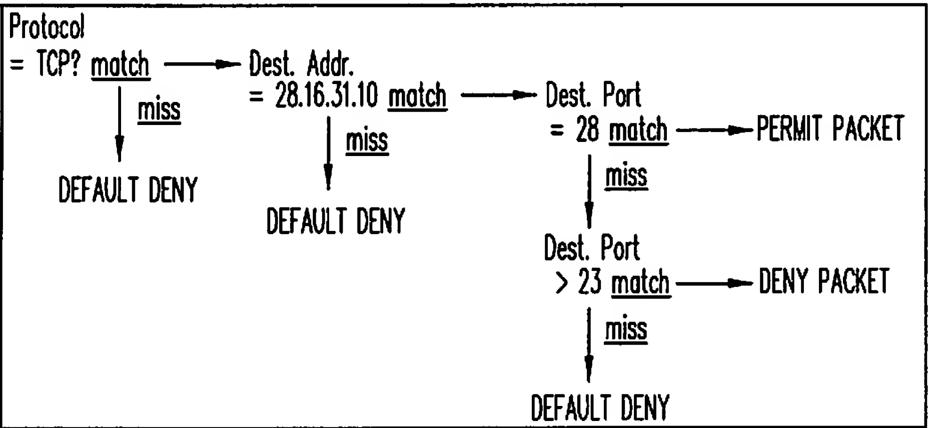
Select bit string constructed from second ACL rule in Rule Set, utilizing the contents of those bit positions (1, 3, 4, and 34) pointed at by the Hash-Table-Balancing Bit Selection Vector, enter hash table at entry corresponding to the bits at bit positions serving as hash key index (e.g., bit position 1 contains "0"; bit position 3 contains "0"; bit position 4 contains "0"; and bit position 34 contains "0") and build binary Comparison Tree indicative of this second selected ACL rule, building on any tree that may already be present for the hash table index.	0000	
	0001	
	0010	
	0011	
	0100	
	0101	
	0110	
	0111	
	1000	
	1001	

FIG. 7D3

Application No.: 09/483,110
First Named Inventor: Faisal Haq
Title: Implementing Access Control Lists Using A Balanced
Hash Table of Access Control List Binary Comparison Trees

	1010	
	1011	
	1100	
	1101	
	1110	
	1111	

FIG. 7D4

Example Showing the Construction of Balanced Hash Table of ACL Binary Comparison Trees (cont.)

Example Showing the Creation of a Binary Comparison Tree for Third In Sequence ACL Rule in Rule Set

Example Showing the Addition of a Binary Comparison Tree Constructed for the Third In Sequence Rule in ACL Rule Set into the Hash Table

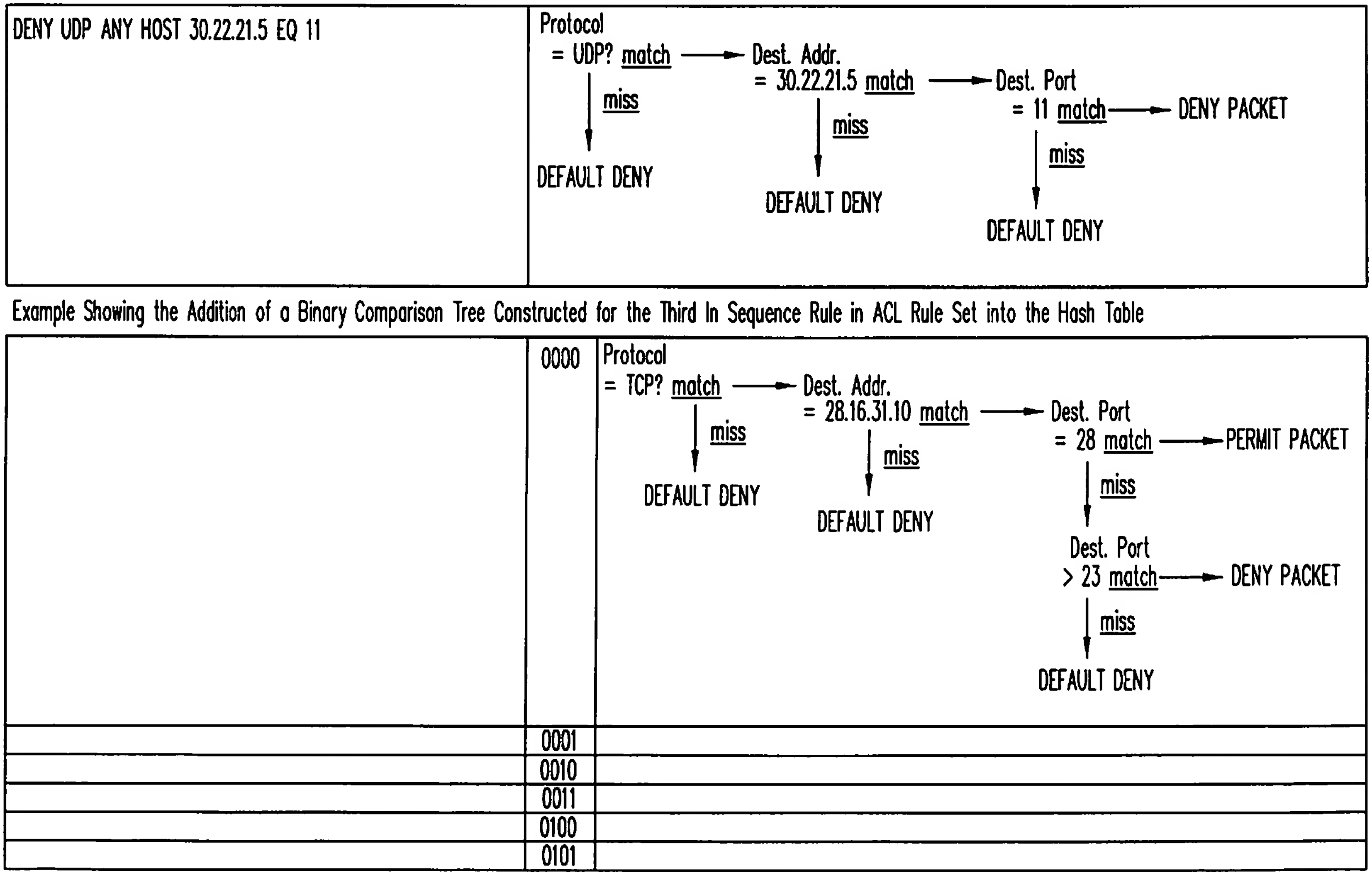


FIG. 7D5

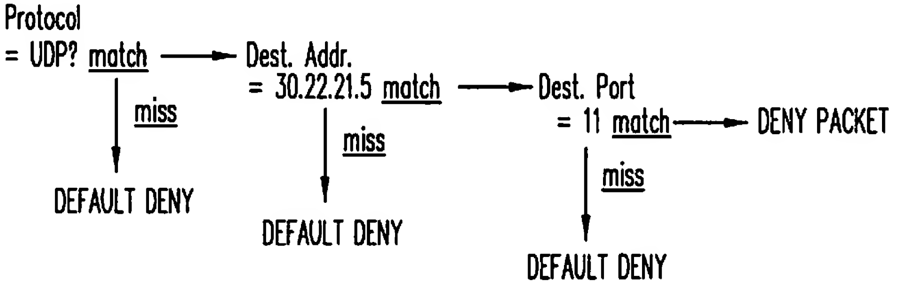
	0110	
	0111	
	1000	
	1001	
	1010	
	1011	
	1100	
	1101	
	1110	
Select bit string constructed from third ACL rule in Rule Set, utilizing the contents of those bit positions (1, 3, 4, and 34) pointed at by the Hash-Table-Balancing Bit Selection Vector, enter hash table at entry corresponding to the bits at bit positions serving as hash key index (e.g., bit position 1 contains "1"; bit position 3 contains "1"; bit position 4 contains "1"; and bit position 34 contains "1") and build binary Comparison Tree indicative of this third selected ACL rule	1111	 Protocol = UDP? match miss DEFAULT DENY Dest. Addr. = 30.22.21.5 match miss DEFAULT DENY Dest. Port = 11 match miss DEFAULT DENY DENY PACKET

FIG. 7D6

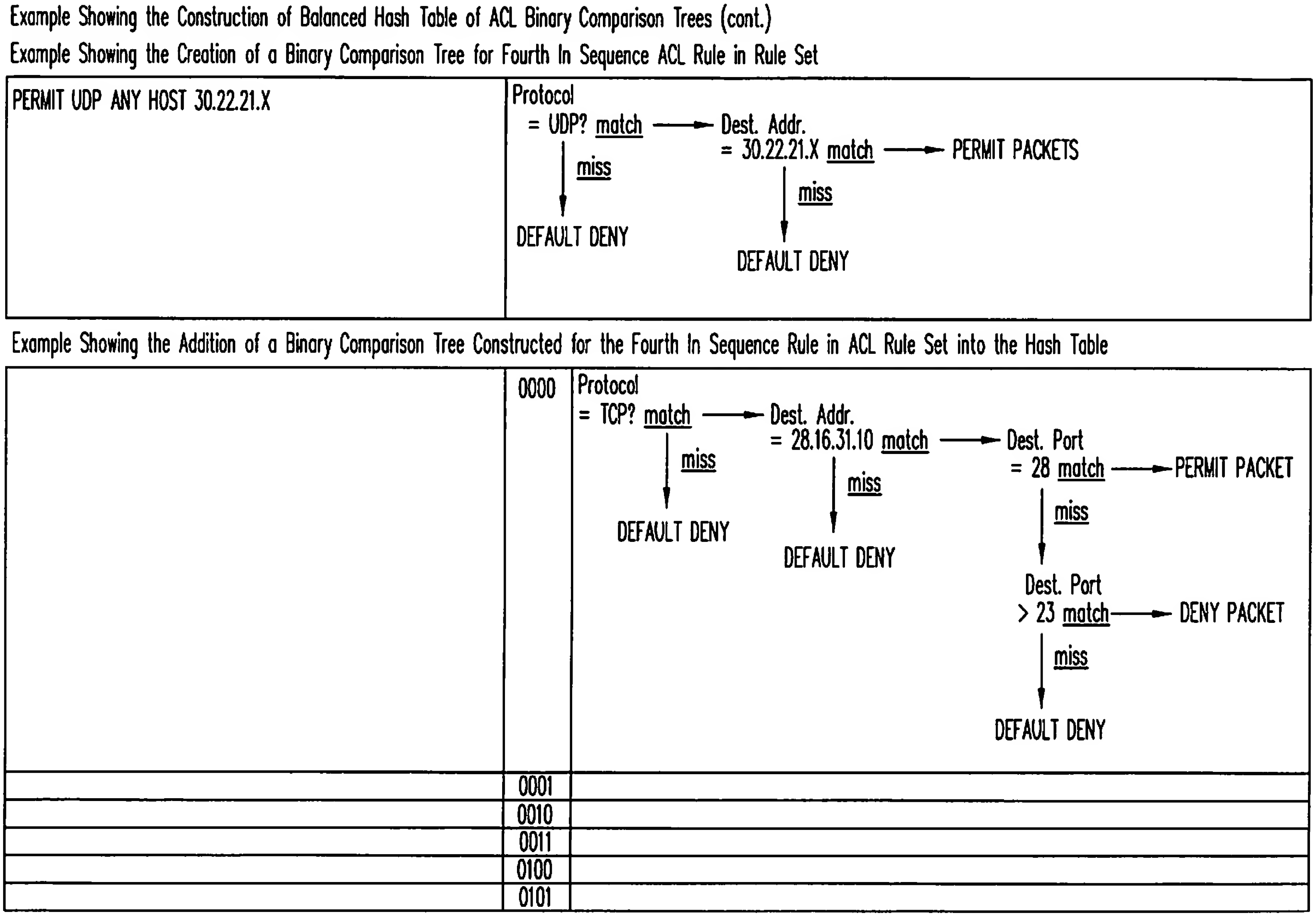

Example Showing the Construction of Balanced Hash Table of ACL Binary Comparison Trees (cont.)
Example Showing the Creation of a Binary Comparison Tree for Fourth In Sequence ACL Rule in Rule Set
PERMIT UDP ANY HOST 30.22.21.X
Protocol
= UDP? match
miss
DEFAULT DENY
Dest. Addr.
= 30.22.21.X match
miss
DEFAULT DENY
PERMIT PACKETS
Example Showing the Addition of a Binary Comparison Tree Constructed for the Fourth In Sequence Rule in ACL Rule Set into the Hash Table
0000
Protocol
= TCP? match
miss
DEFAULT DENY
Dest. Addr.
= 28.16.31.10 match
miss
DEFAULT DENY
Dest. Port
= 28 match
PERMIT PACKET
miss
Dest. Port
> 23 match
DENY PACKET
miss
DEFAULT DENY
0001
0010
0011
0100
0101

FIG. 7D7

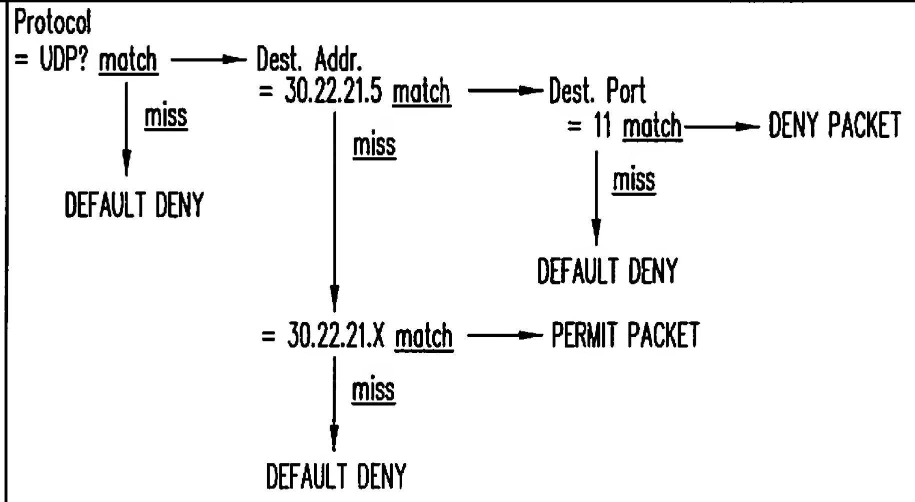
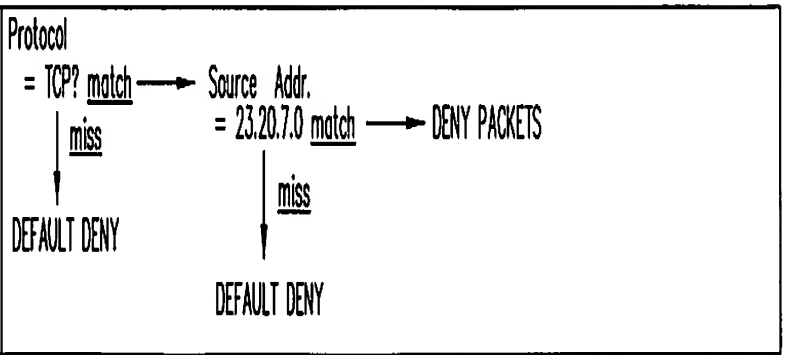
	0110	
	0111	
	1000	
	1001	
	1010	
	1011	
	1100	
	1101	
	1110	
Select bit string constructed from fourth ACL rule in Rule Set, utilizing the contents of those bit positions (1, 3, 4, and 34) pointed at by the Hash-Table-Balancing Bit Selection Vector, enter hash table at entry corresponding to the bits at bit positions serving as hash key index (e.g., bit position 1 contains "1"; bit position 3 contains "1"; bit position 4 contains "1"; and bit position 34 contains "1") and build binary Comparison Tree indicative of this fourth selected ACL rule, building on any tree that may already be present for the hash table index	1111	 Protocol = UDP? match miss DEFAULT DENY Dest. Addr. = 30.22.21.5 match miss Dest. Port = 11 match DENY PACKET miss DEFAULT DENY = 30.22.21.X match PERMIT PACKET miss DEFAULT DENY

FIG. 7D8

Example Showing the Construction of Balanced Hash Table of ACL Binary Comparison Trees (cont.)

Example Showing the Creation of a Binary Comparison Tree for Fifth In Sequence ACL Rule in Rule Set

DENY TCP 23.20.7.0 X.X.X.X.	

Example Showing the Addition of a Binary Comparison Tree Constructed for the Fifth In Sequence Rule in ACL Rule Set into the Hash Table

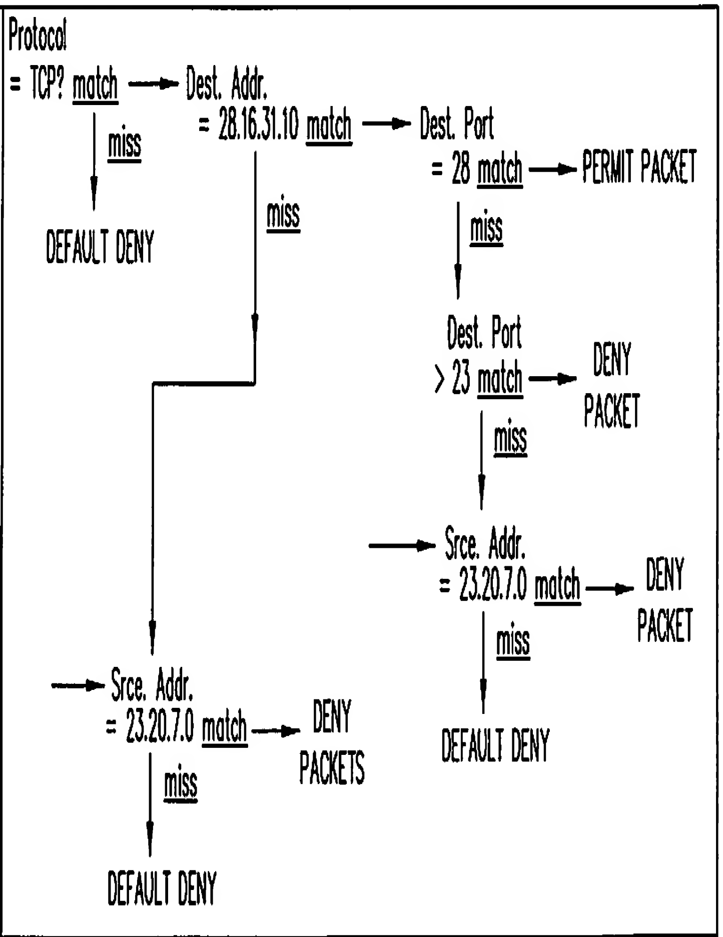
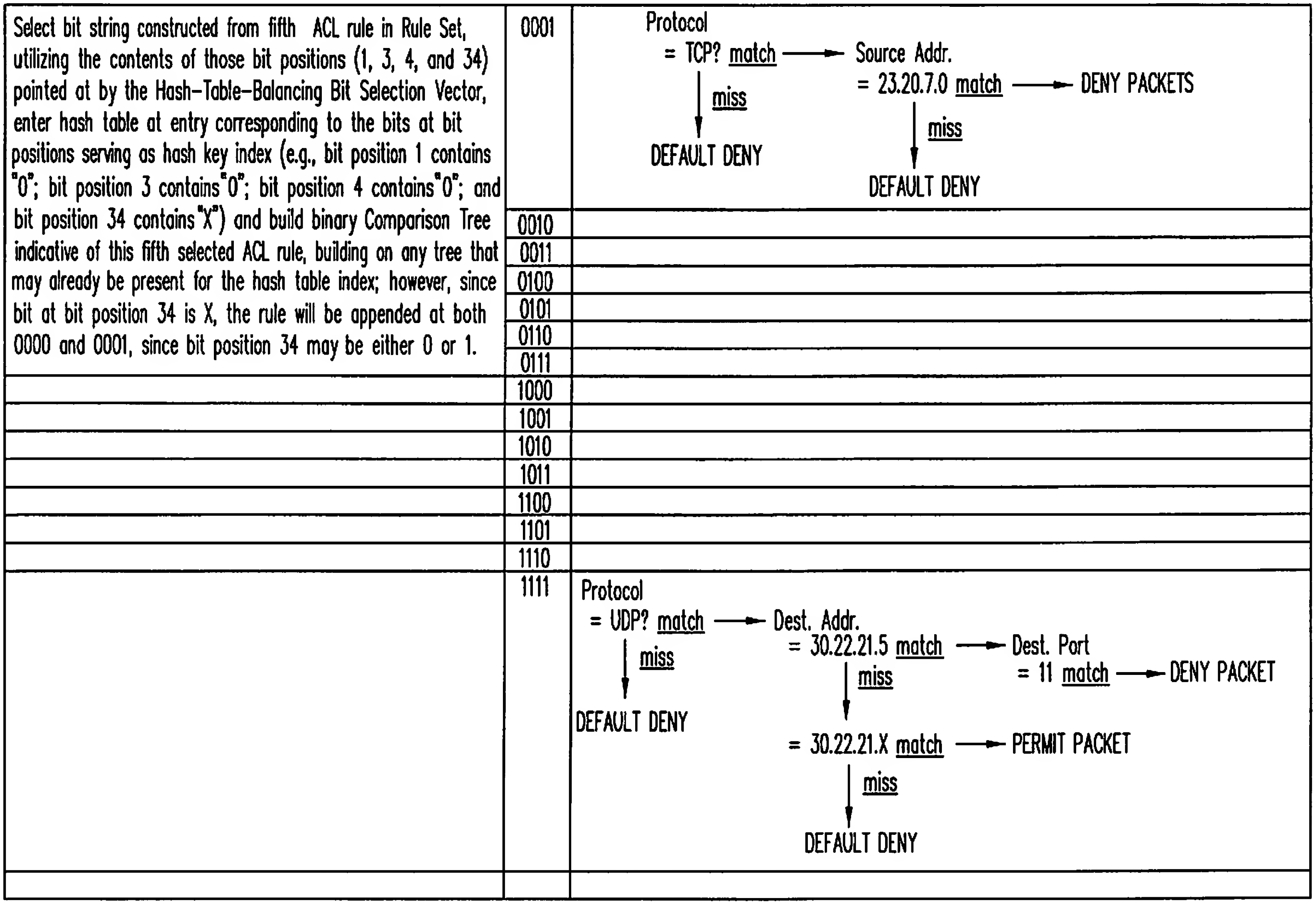
Select bit string constructed from fifth ACL rule in Rule Set, utilizing the contents of those bit positions (1, 3, 4, and 34) pointed at by the Hash-Table-Balancing Bit Selection Vector, enter hash table at entry corresponding to the bits at bit positions serving as hash key index (e.g., bit position 1 contains "0"; bit position 3 contains "0"; bit position 4 contains "0"; and bit position 34 contains "X") and build binary Comparison Tree indicative of this fifth selected ACL rule, building on any tree that may already be present for the hash table index; however, since bit at bit position 34 is X, the rule will be appended at both 0000 and 0001, since bit position 34 may be either 0 or 1. In addition, since the rule itself applies to any destination address, the miss branch of all destination branches present must feed back into the source address compare instruction associated with this Fifth Rule.	0000	

FIG. 7D9

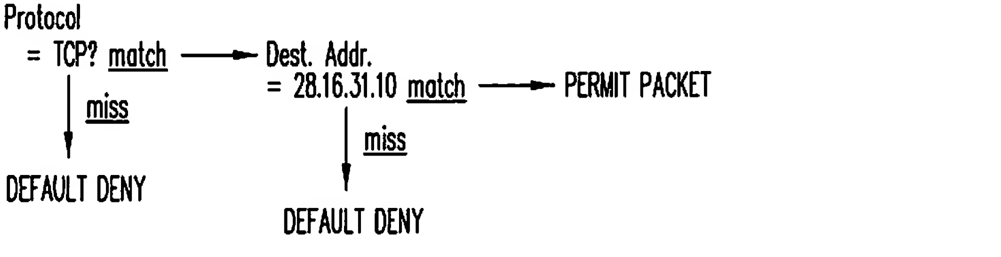


Select bit string constructed from fifth ACL rule in Rule Set, utilizing the contents of those bit positions (1, 3, 4, and 34) pointed at by the Hash-Table-Balancing Bit Selection Vector, enter hash table at entry corresponding to the bits at bit positions serving as hash key index (e.g., bit position 1 contains "0"; bit position 3 contains "0"; bit position 4 contains "0"; and bit position 34 contains "X") and build binary Comparison Tree indicative of this fifth selected ACL rule, building on any tree that may already be present for the hash table index; however, since bit at bit position 34 is X, the rule will be appended at both 0000 and 0001, since bit position 34 may be either 0 or 1.	0001	Protocol = TCP? match → Source Addr. = 23.20.7.0 match → DENY PACKETS TCP? miss → DEFAULT DENY 23.20.7.0 miss → DEFAULT DENY
	0010	
	0011	
	0100	
	0101	
	0110	
	0111	
	1000	
	1001	
	1010	
	1011	
	1100	
	1101	
	1110	
	1111	Protocol = UDP? match → Dest. Addr. = 30.22.21.5 match → Dest. Port = 11 match → DENY PACKET UDP? miss → DEFAULT DENY 30.22.21.5 miss → = 30.22.21.X match → PERMIT PACKET 30.22.21.X miss → DEFAULT DENY

FIG. 7D10

Example Showing the Construction of Balanced Hash Table of ACL Binary Comparison Trees (cont.)

Example Showing the Creation of a Binary Comparison Tree for Sixth In Sequence ACL Rule in Rule Set

PERMIT TCP ANY HOST 28.16.31.10	

Example Showing the Addition of a Binary Comparison Tree Constructed for the Sixth In Sequence Rule in ACL Rule Set into the Hash Table

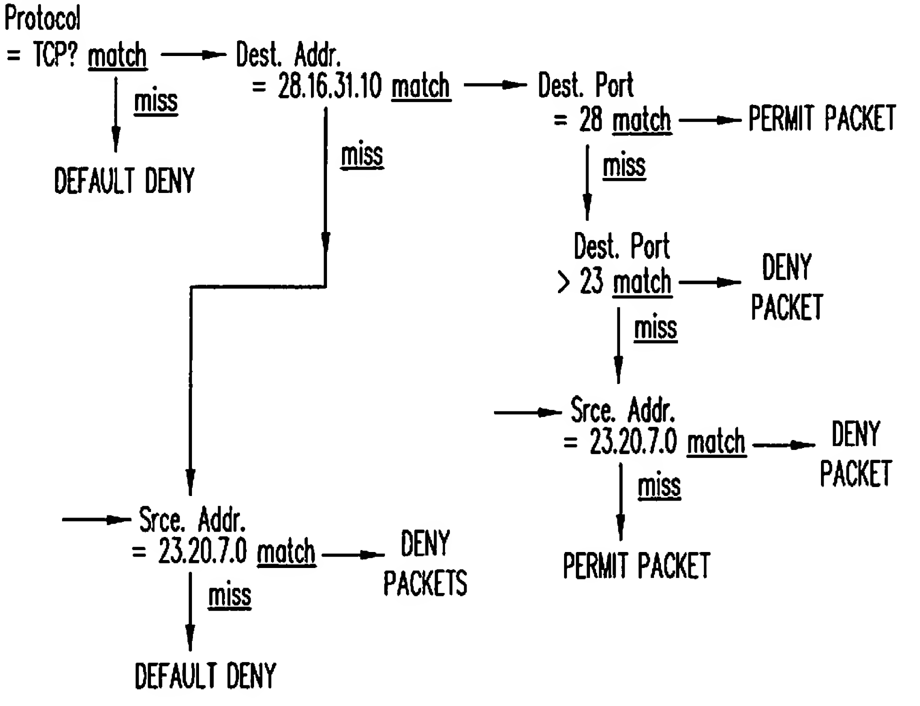
Select bit string constructed from sixth ACL rule in Rule Set, utilizing the contents of those bit positions (1, 3, 4, and 34) pointed at by the Hash-Table-Balancing Bit Selection Vector, enter hash table at entry corresponding to the bits at bit positions serving as hash key index (e.g., bit position 1 contains "0"; bit position 3 contains "0"; bit position 4 contains "0"; and bit position 34 contains "0") and build binary Comparison Tree indicative of this sixth selected ACL rule, building on any tree that may already be present for the hash table index.	0000	

FIG. 7D11

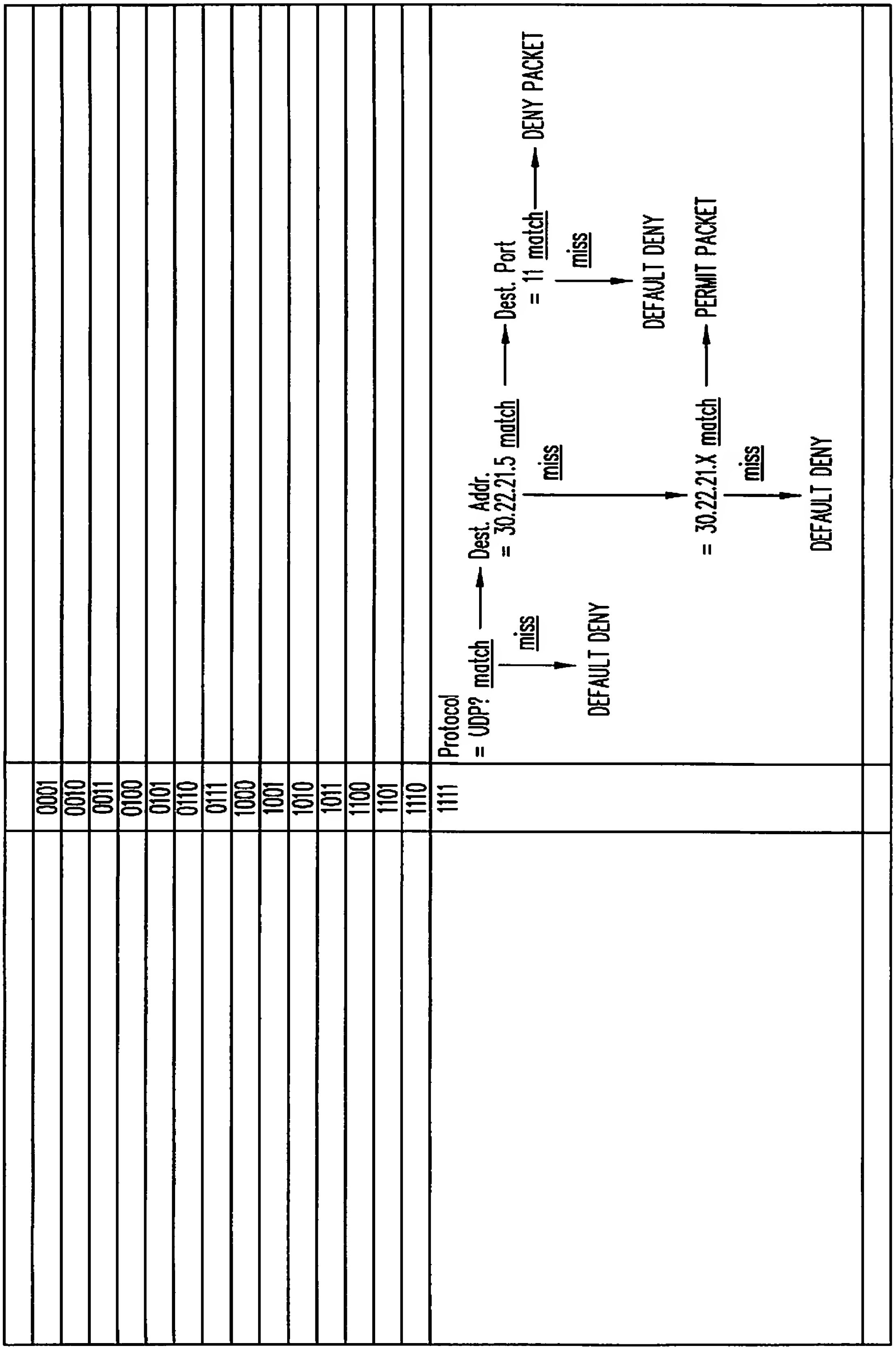

0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
Protocol
= UDP? match
miss
DEFAULT DENY
Dest. Addr.
= 30.22.21.5 match
miss
Dest. Port
= 11 match
miss
DEFAULT DENY
DENY PACKET
= 30.22.21.X match
miss
DEFAULT DENY
PERMIT PACKET

FIG. 7D12